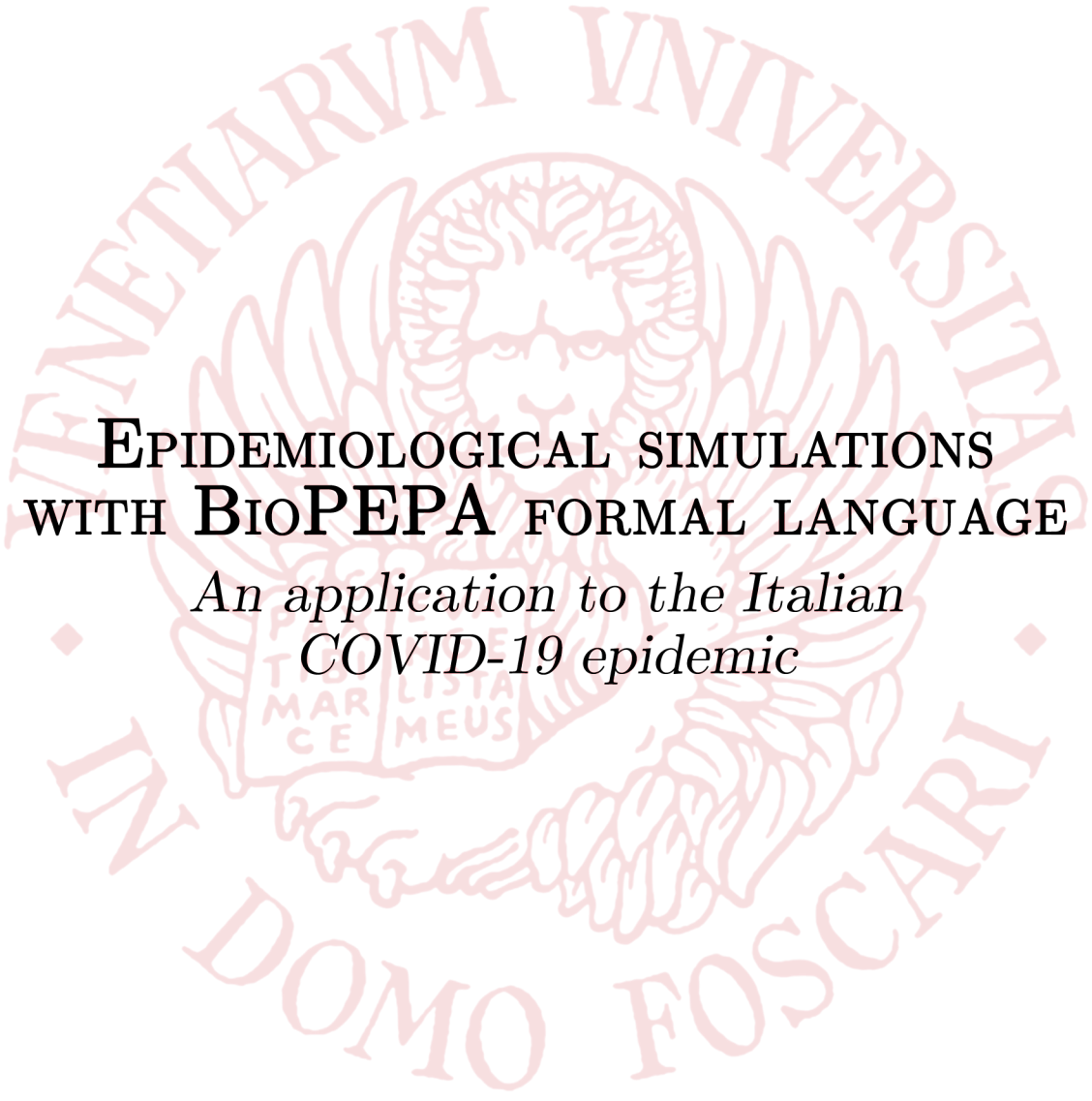
Università Ca' Foscari - Venezia
Dipartimento di Scienze Ambientali, Informatica e Statistica

Corso di Laurea in

Computer Science
Software Dependability and Cyber Security



Epidemiological simulations with BioPEPA formal language

An application to the Italian COVID-19 epidemic

Relatore:
Prof. Damiano Pasetto

Laureando:
Roberto Perissa mat. 859143

Co-relatore:
Prof. Andrea Marin

Anno Accademico 2021/2022

Contents

Abstract

Epidemiological models played a crucial role during SARS-CoV-2 pandemic to assist policy-maker authorities in the roll-out of non-pharmaceutical interventions such as lockdowns, limitations to individuals mobility, mask wearing and vaccination campaigns.

This thesis explores the possibility of using BioPEPA to simulate real epidemics which are subject to complex dynamics (due, for example, to a heterogeneous age structure of the population, vaccination campaigns, temporal changes in the interactions among agents). The formal language BioPEPA describes the interactions among the agents that constitute the population. From this formalism it derives the compartmental models (stochastic or deterministic) associated to the mean-field solution. In order to achieve this objective, this thesis developed new scripts to produce the input files for the BioPEPA workbench when considering complex epidemiological models and to analyse the BioPEPA outputs. To prove the potential of the developed software to analyse real epidemics, the thesis presents an application to the Italian COVID-19 epidemic. In particular, the simulations will focus on estimating the impact of the Italian vaccination campaign that started in December 2020, where priority was given to healthcare workers and elderly individuals, having a higher risk of serious symptoms and death. This study aims to show how objective model simulations can support risk evaluation.

Chapter 1

Introduction

The globalisation phenomenon that in the last decades contributed to trigger the ongoing climate change and the consequent natural disasters that poses many challenges to the human civilisation. Many of these natural disasters are generated by the unbalancing of local and then global ecosystems, from physical, like wind currents, to biological, like mass migration of animals. A very sneaky problem that humanity is facing is the emerging of new pathogens and diseases caused by humans interference in animal habitats and vice versa, for example for the increase of events in which people come in contact with different, rare or unusual species that before were isolated in wilderness[1, 2, 3, 4]. Biomedical scientists have warned against this underrated threat since the early 2000s[5] but the threat was not widely acknowledged both by authorities and by the general population[6].

In this context the global population was caught unprepared when the new SARS-CoV-2 virus emerged in China in late 2019 and quickly spread all over the globe in few weeks, posing incredible challenges to safeguard human lives and the economic stability. The WHO declared the pandemic status on the 11th of March 2020 [7], urging the scientific community and governments to increase the efforts to contain the transmission and develop a vaccine for the virus.

The first few batches of Covid-19 vaccines had begun to arrive in Italy in December 2020 and they had been reserved to healthcare workers. In the following month the delivering pace increased at industrial scale and the vaccination campaign could actually begin. The administration of doses at the nation scale is a difficult task which involves both logistical and health policy problems. In particular, decision makers chose to prioritise the vaccination of elderly people and selected individuals with immune-compromised situations[8]. This policy is based on the assumption that if the prioritised recipients of the doses contract the disease without the vaccine protection they would be the most likely to suffer serious consequences and possibly

die. This policy is correct and pragmatic under the logic of minimizing health and life threatening at population level. Moreover, in the first months of 2021 the elderly and the scholar-aged population were under movement restrictions, only working-age people were moving, thus it seemed possible that the vaccine-induced immunization could also prevent the spreading of the virus. However, under the general target of halting the spreading of the Covid-19 epidemic the adopted vaccination policy might be just one of the possible ways of actions[9, 10].

How would the pandemic change if the vaccination policy would have prioritised mobile people instead of the most fragile? Is there a vaccination strategy that would have stop the epidemic earlier? What other scenarios could have happen? Answering to these questions is a problem belonging in the epidemiology field.

Many studies have been made with different approaches and objectives, each expanded the knowledge on the disease object of study and/or techniques to increase the details an epidemiological model is able to handle and give insight on. One of the first epidemiological studies modelled and analysed the spatial spreading of Cholera among a population, investigating the possible mean of infections [11, 12]. This work lead to more advanced medical researches and then to some of the most well know sanitization actions and infrastructures for personal and public health that are given for granted nowadays i.e. sewers.

Give these many year of experience in epidemiological studies, when the SARS-CoV-2 pandemic hit in almost every country in the World, the scientific community started to analyze many aspects of the ongoing disease spreading. Few weeks after the beginning of the Italian epidemic, *Cereda et al* (2021), published an analysis of the transmission dynamics and the geographical diffusion of Covid in Italy[13]. As weeks and months passed more accurate and deeper studies were made on the effectiveness of restrictive measures (like quarantine) and implications of the policies removal[14, 15, 16]. Finally, when vaccines were approved the logistical problem of their distribution made the case for interesting investigations on how their distribution could affect the epidemiological situation[17, 9].

The purpose of the work presented in the next chapters is to provide a step forward in the construction of epidemiological models for the simulation of an epidemic.

The formalism chosen in this thesis to describe the virus transmission in a population is the one provided by the BioPEPA language[18]. BioPEPA is a *Performance Evaluation Process Algebra* developed to represent biochemical reactions and systems. BioPEPA model definitions start from the basic interactions among the agents to then simulate the large-scale epidemiological dynamics through different

types of stochastic or deterministic solvers. BioPEPA offers two tools, the BioPEPA Eclipse Plugin and the BioPEPA Workbench, that allow to simulate the models and perform other operations. The simulation algorithms available in the Workbench are Gillespie's SSA algorithm implemented in the StochKit library and the additive Runge-Kutta method with Sundials library[19]. The Eclipse Plugin provide Gillespie's stochastic simulation algorithm, Gillespie's Tau-Leap stochastic simulation algorithm, Gibson-Bruck stochastic simulation algorithm and Runge-Kutta methods algorithms[20].

This project focuses on the stochastic (StochKit library which implements Gillespie Stochastic Simulation Algorithm (SSA) Direct Method)[21] and deterministic (Ordinary Differential Equations) solvers (Sundials suite [22]) of the BioPEPA tool suite, giving the opportunity to have both kinds of solvers applied automatically to the same model. Furthermore the formalization in BioPEPA allows for further work for formal model checking. The implementation work of this project target the improvement of the construction of the epidemiological model files, the introduction of parameters changing in time in the simulation for the reproduction of realistic epidemics and the analysis of the outputs of simple models for comparison with the available epidemiological data. The models proposed are variations of the well known SIR (Susceptible, Infected, Recovered) model. The epidemiological description of the population is considered both in classless and classed models scenarios, i.e. the population is considered uniform or is considered subdivided in subsets. The classes can be introduced in the models to account for the different characteristics of the population and to study their relevance on the epidemic evolution. Lastly the models are upgraded with the vaccination campaign.

The final aim of this work is that the tools, models and results proposed may help gaining knowledge on epidemics behaviour and inspire future research. The development of a more comprehensive software, possibly exploiting OOP (Objective Oriented Programming) techniques, that would greatly expand the simulation capabilities through BioPEPA and hoping that this could help decision makers during an health emergency and other similar cases.

Chapter 2

Epidemiological models and BioPEPA

The first part of this chapter presents a brief introduction to epidemiological modelling, with particular focus to the mathematical definition of the models adopted in this thesis. The second part of the chapter presents the BioPEPA formal language alongside with the reasons why to adopt this tool in modelling an epidemic.

2.1 Epidemiological models

Epidemiological models are mathematical models used to reconstruct and project the evolution of an epidemic and its likely outcome. They are crucial for medical research as well as for various authorities in their policy making decisions. The most used and well know kind of models are *Compartmental Models*, which can be applied with various techniques. Their name derive from their structure: compartmental models divide the population in compartments, and each compartment describes the infectious status of an individual and possibly other information. The total sum of individuals in each compartments gives the complete population, it can be assumed constant in time, under the assumption of an equilibrium among birth and mortality rates, and no immigration/emigration processes. Each individual in the population might become infected and then move between compartments, often this passage is unidirectional. Generally, the sequence and compartments names are resumed in the model name. For example the most common model is SIR model, composed of three compartments:

- S: number of susceptible individuals. When a susceptible and an infectious individual come into contact, the susceptible individual contracts the disease and transits to the infectious compartment;

- I: number of infectious individuals. These are individuals who have been infected and are capable of infecting susceptible individuals;

- R: number of recovered and now immune or deceased individuals.

The passage of individuals among compartments, or better the variation of the number of individuals in each compartment, can be easily described by a system of ordinary differential equations, ODEs, under the assumption that the population can be approximation with continuous (and not discrete) numbers. The solution of these equations through a numerical integrator (e.g. based on a Runge-Kutta scheme) provides a deterministic solution to the problem.

This approach is largely applied in epidemic studies because ODEs are intrinsically descriptive of the dependency relation of every compartment to the evolution[23]. This behaviour is straightforward modelled with time dependant rates that resume the processes between compartments, see for example the 2.2 section.

Another possible mean of simulating epidemiological models is with a stochastic process[24]. Stochastic processes are a mathematical object, formally defined, that uses probability distributions of random variables to model action times inside systems that appear to have a random behaviour. Among the most popular processes there are the Markov chains and Gillespie algorithm. Markov chains[25] are a popular simulation tool because they support both discrete and continuous time sequences and are applied to many different fields. The main property of the Markov chains is that the next status of the system is dependant only on the current status and events probabilities. This process is independent from previous process values. Gillespie algorithm[26] instead is a simulation tool which is very popular in the biochemical field and it is based on a progression logic. At each time t all system variables are evaluated with explicit simulation, then the next time $t + \Delta t$ (with time step Δt) status is set from the t status and the loop is repeated till the termination condition.

The subdivision of the population into classes is one of the main information that is possible to add to the compartments in more complex models. This subdivision represents specific information about the population and it provides extra insights in the epidemic evolution. Examples of possible classes are sex, age, work category or preexisting health conditions[27, 28, 29]. Adding these classes in the model implies the description of more complex transmission processes and the use of more

advanced mathematical tools. The *basic reproduction number*, denoted by R_0, is the expected number of secondary cases directly generated by one infected individual in a population where all individuals are susceptible. This factor is not a constant value, since it depends on different aspects like environmental temperature, specific number of contacts, transmissivity and susceptibility of individuals in the different population classes and more. However, it's possible to compute an average for each disease and thus give a common reference for each considered strain. For example, the R_0 of the first strain of Covid-19 is equal to 2.87[30].

Differently, the *effective reproduction number* is the number of secondary cases directly generated by one infected individual among the susceptible individuals at time t population. Hence, it varies during an epidemic and it is noted with R_t. Expecting that a disease could spread constantly with the R_0 factor in an evolving scenario is not plausible. The unavoidable variation is that, during the disease transmission, the population itself is no more consisting of susceptible individuals only because of the following factors: a certain level of disease-induced or vaccine-induced immunity, behavioural changes due to awareness of possible transmission, measures taken by the health authorities against spreading of the disease.

The value of R_t tells if the number of cases will increase (if $R_t > 1$), decrease (if $R_t < 1$), or remain stable (if $R_t = 1$), indicating that the disease is becoming endemic. The effective reproduction number is a key element in epidemiology: many epidemiological models can estimate its value from the computation of the strength of the infection, which tells how many individuals are infected in a certain moment.

In ODE-based models, the effective reproduction number is directly linked to the stability of the *disease free equilibrium*. It is defined as the state of an epidemic in which the number of infected individuals is zero, implying the end of the transmission (in a closed system). In simpler words, this equilibrium is stable if a small perturbation (e.g., the presence of an infected individuals) is rapidly re-absorbed (i.e., the new infections decrease in time), thus avoiding a new outbreak. The condition $R_t < 1$ implies the stability of the disease-free equilibrium.

The estimation of Rt in complex epidemiological models entails the computation of the so-called Next Generation Matrix (NGM, see [31]). A multi-class model typically describes the evolution of different compartments for each class subject to different rates. This is often represented with matrices, which also simplify the mathematical computations. Seeing the infections process over time as a sequence of generations of infected individuals, it is possible to describe the transmission at a time instance through the evaluation of the *Next Generation Matrix* which is here denoted with $K(t)$. The NGM is defined as the matrix which relates the numbers of

newly infected individuals in the various categories in consecutive generations. The $K(t)$ matrix is computed through of two other matrices: $K = T + \Sigma$ where T is a matrix representing the `transmission` part and Σ the `transition` part, NGM is the matrix $-T * \Sigma^{-1}$, and R_t is its spectral radius (largest positive eigenvalue).

2.1.1 From compartmental models to agent based models

The process algebra that is selected in this project for the formalisation of the models has a different perspective as compared to the compartmental-based model definitions. BioPEPA [18] describes the population as a set of agents that interact among each other through a sequence of actions and possibly change status in time, meanwhile the classic compartmental-based models count the number of individuals in each compartment and make them change status with a global perspective.

Thus, to define an epidemiological model with BioPEPA equivalent to a classic ODE, it is necessary to compute rates such that they represent state transitions of a single agent instead of a whole compartment population per unit of time. Depending on the data available, the actions and rates might be easier or harder to describe from the agent or the global perspective.

Using R_t to reproduce the epidemic

A major problem in epidemiological models is to reproduce the trend and temporal changes of the force of the infection. This is particularly important for an epidemic such as Covid19, where government-imposed restrictions in human mobility, the use of masks and vaccination policies drastically changed the transmission dynamics in few months. These changes in transmission might correspond to temporal changes in the model parameters that are difficult to track and estimate from the data.

In this thesis a different approach is adopted to obtain model results that are qualitatively comparable with the data. As already stated, the key element representing the trend of the epidemic is the R_t factor, which is typically estimated during the course of the epidemic by health authorities. and this proves to be useful because of the following property of the NGM matrix. The R_0 factor is the dominant eigenvalue of K, this allows computation to obtain the R_0 value from a model or vice versa to obtain the K matrix, thus the force of infection, from R_0 or R_t. The replication of infection trend will be performed through the R_t factor, allowing a stronger control over the scenario evolution. This integration grants better reliability for the base scenario in which the vaccination action can be included with varying rates to study possible outcomes.

2.2 SIRDV model

The SIRDV model is the most simple compartmental model to simulate an epidemic having a relevant number of deceased individuals. For this reason, it is the first model proposed in this thesis to simulate the SARS-CoV-2 virus epidemic. The SIRDV model is based on the well-known SIR model and it substantially adds the Death compartment to track the severe consequences of the infection and the Vaccinated compartment to track the immunization campaign and its effects on the epidemic. This model simplifies the real life transmission of the epidemic imposing the following assumptions:

- the population is stable, i.e. birth and deaths processes are neglected the number of newborns are approximately the same of natural occurring deaths (except for the deaths directly linked to the disease);
- the population is considered a well-mixed and closed system without imported/exported cases;
- individuals are infectious as soon as they contact the virus and they do not isolate due to quarantine measures;
- no individual-specific behaviours are represented, like the difference of ineffectiveness between symptomatic and asymptomatic individuals;
- recovered individuals (individuals that survived after the infection) have a permanent natural immunity to the infection, and thus no reinfection is possible by the same virus or by a variant of the disease. This assumption is particularly weak when simulating an epidemic for a long period, since immunity might wane and reinfections are possible (especially under new variants, for example what happened with variant Omega);
- vaccines are distributed to S and R compartments;
- like recovered individuals, who have received the vaccine is permanently immune and cannot become infected;
- only one dose of vaccine is sufficient for fully immunization of the individual.

These assumptions simplify the complex transmission dynamics of the disease in the population, however the model is still useful to provide overall insights on the trend of the epidemic and estimate the impact of control measurements, such as the vaccination campaign.

In particular, it is possible to estimate the temporal changes in the model parameters in order to follow the infection waves and the total number of recovered

and dead people in the system.

2.2.1 SIRDV model equations

The SIRDV model subdivides the population of a system (N) into four different compartments, schematised in figure2.1, characterised by different stages of the infection:

- S: compartment of the susceptible individuals, indicates the individuals that do not have any immunity and can become infected. When a new disease hits a community, all the individuals are susceptible except for the first infected individual (starting condition of the model);
- I: Infected compartment, this compartments counts the number of individuals that are infected at a given time, both symptomatic and asymptomatic. These individuals contribute to pass the pathogen to the susceptible agents;
- R: Recovered compartment, in this state arrive all individuals that have survived the disease or that may have been immunized. In the model considered in this study, individuals in R cannot be reinfected do not lose the immunization;
- D: Dead state, all agents that unfortunately died during the disease arrive in this compartment;
- V : Vaccinated state, in this state arrive all agents that have been immunized against the disease.

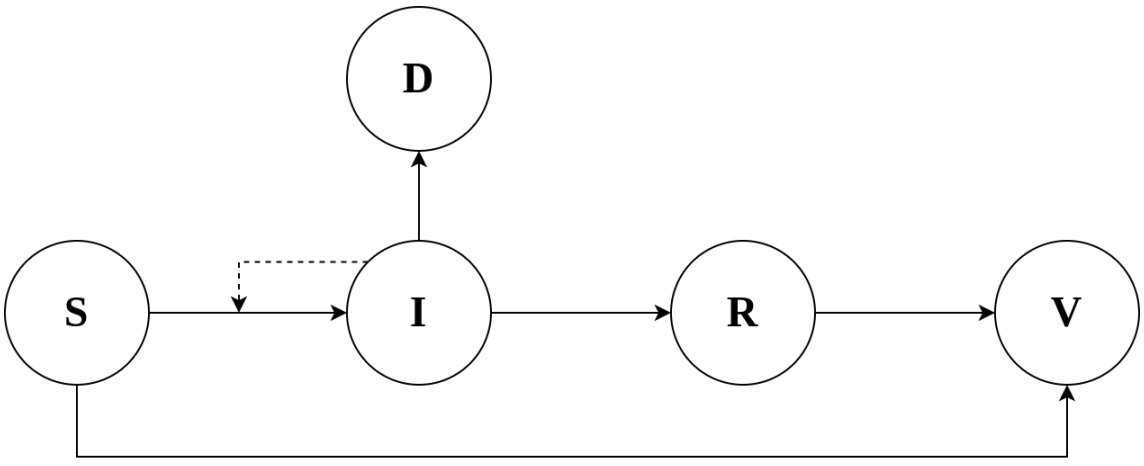


Figure 2.1: Transition scheme of SIRDV model

Assuming that the total population at any time t is N, $N = S(t) + I(t) + R(t) + D(t) + V(t)$, the ordinary differential equations (ODEs) that describe the transition

of the individuals in the model are

$$\begin{cases} \dot{S} = -\beta \frac{I}{S+I+R+V} S - \delta_S S = -(\lambda + \delta_S) S \\ \dot{I} = \beta \frac{I}{S+I+R+V} S - (\alpha + \gamma) I \\ \dot{R} = \alpha I - \delta_R R \\ \dot{D} = \gamma I \\ \dot{V} = \delta_S S + \delta_R R \end{cases} \tag{2.1}$$

where the parameters represent mean permanence time in the compartments. The term λ is called the instantaneous force of the infection and represents the percentage of new cases among the susceptible population in a certain instant:

$$\lambda = \frac{\beta I}{S + I + R} \tag{2.2}$$

The α, γ and δ are the recovery, death and vaccination rate respectively. The vaccination rate is subdivided between the susceptible and recovered states, represented by δ_S and δ_R in such a way that $\delta_S * S + \delta_R * R$ corresponds to the total vaccines deployed per temporal unit.

This system of ODEs is typically initialized using the following initial state: $S(0) = N - I(0)$; $I(0) = 1$ while all the other states are set to 0 at the beginning.

2.2.2 Basic reproduction number

We recall here how to compute the basic and effective reproduction numbers for the SIRDV model as leading eigenvalues of the Next Generation Matrix (the complete computations are in [31])

To compute the R_0 factor related to the SIRDV model it is necessary to linearize the non-linear ODEs related to the infectious compartments around the disease free equilibrium.

In this case we consider only the equation for the variation of I, $\dot{I} = f(I)$ where

$$f(I) = \beta \frac{SI}{S + I + R + V} - (\alpha + \gamma) I$$

Clearly $I(t) = 0$ is a constant solution of this equation, and to evaluate if this solution is stable we need to linearize the system:

$$\frac{df}{dI}(I) = \beta\frac{S}{S+I+R+V} - \beta I\frac{S}{(S+I+R+V)^2} - (\alpha+\gamma)$$

Then, evaluating at the disease free equilibrium, $I = 0$ we obtain;

$$\begin{aligned}\frac{df}{dI}(0) &= \beta\frac{S}{S+R+V} - (\alpha+\gamma) = \\ &= T + \Sigma\end{aligned}$$

where $T = \beta\frac{S}{S+R+V}$ is the transmission matrix (in this model T is a number and it indicates the linearized number of new infections generated by one infectious individual) and $\Sigma = -\alpha - \gamma)$ is the transition matrix (in this model Σ is a number and it indicates the mean time spent by an individual in the I compartment).

The next generation matrix then is defined as $NGM = -T\Sigma^{-1}$ and R_t (or R_0) is its largest positive eigenvalue (spectral radius). In the SIRDV model, the NGM matrix is a number and R_t is

$$R_t = \rho(-T\Sigma^{-1}) = \frac{\beta}{\alpha+\gamma}\frac{S}{S+R+V}$$

At the beginning of the epidemic, when the population is completely susceptible, we have $R = V = 0$, and the basic reproduction number is:

$$R_0 = \rho(-T\Sigma^{-1}) = \frac{\beta}{\alpha+\gamma}\frac{S}{S+R+V}$$

2.2.3 Multi class model equations

The SIRDV model can be easily extended to a more complex case in which the population is subdivided into classes, e.g., age classes. Each class might be identified by suitable model parameters (susceptibility, transmissivity, mortality) and the virus might be transmitted among classes through an estimate of the contact matrix.

Assuming that there are M classes, the total population at any time t is

$$N = \sum_{i=1}^{M} S_i(t) + I_i(t) + R_i(t) + D_i(t) + V_i(t);$$

The ODEs for a multi class approach and for each class i are similar to the ones of

eq. (2.1):

$$\begin{cases} \dot{S}_i = -(\lambda_i + \delta_{S_i})S_i \\ \dot{I}_i = \lambda_i S_i - (\alpha_i + \gamma_i)I_i \\ \dot{R}_i = \alpha_i I_i - \delta_{R_i} R \\ \dot{D}_i = \gamma_i I_i \\ \dot{V}_i = \delta_{S_i} S_i + \delta_{R_i} R_i \end{cases} \tag{2.3}$$

Where now the force of infection is class-specific and takes into account the possible contacts among individuals in different classes:

$$\lambda_i = \beta \frac{\eta_i \sum_{j=1}^{M} \tau_j c_{ij} I_j}{\sum_{j=1}^{M} c_{ij} S_j + I_j + R_j + V_j} \tag{2.4}$$

where c_{ij} represents the daily contacts of an agent of class i with agents in class j; η_i is the susceptibility of class i; and τ_j is the transmissibility of infected in class j.

Also in this case R_t can be evaluated through the spectral radius (ρ) of the NGM matrix,

$$R_0 = \rho(-T\Sigma^{-1})$$

where the transmissivity matrix T is

$$T = \beta \begin{pmatrix} \eta_1 c_{11}\tau_1, & ..., & \eta_1 c_{1n}\tau_n \\ \eta_2 c_{21}\tau_1, & ..., & \eta_2 c_{2n}\tau_n \\ ... & ... & ... \\ \eta_n c_{n1}\tau_1, & ..., & \eta_n c_{nn}\tau_n \end{pmatrix} = \beta\tilde{T}$$

and the transition matrix Σ is:

$$\Sigma = \begin{pmatrix} -\alpha_1 - \gamma_1, & 0, & ..., & 0 \\ 0, & -\alpha_2 - \gamma_2, & ..., & 0 \\ 0, & ..., & ..., & 0 \\ 0, & ..., & ..., & -\alpha_n - \gamma_n \end{pmatrix}$$

$$-\Sigma^{-1} = \begin{pmatrix} \frac{1}{\alpha_1+\gamma_1}, & 0, & ..., & 0 \\ 0, & \frac{1}{\alpha_2+\gamma_2}, & ..., & 0 \\ 0, & ..., & ..., & 0 \\ 0, & ..., & ..., & \frac{1}{\alpha_n+\gamma_n} \end{pmatrix}$$

Thanks to this computation, it is possible to link the temporal changes on R_t

(e.g., obtained from the data) to changes in the parameter β (assuming that the other parameters do no change in time). Thus

$$\rho(-T\Sigma^{-1}) = R_t$$
$$\beta\rho(-\tilde{T}\Sigma^{-1}) = R_t$$
$$\beta = \frac{R_t}{\rho(-\tilde{T}\Sigma^{-1})}$$

2.3 SEIRDV model

The SEIRDV model is the second model proposed in this thesis to simulate an epidemic. It is based on the previusly presented SIRDV model, with the addition of a compartment for the exposed individuals (E). All the assumptions of the SIRDV model are still valid in the SEIRDV. The additional compartment of exposed individuals is used to represent the time period needed to incubate the disease and begin to show symptoms. The individuals in the exposed compartment are not infective, so the infection action is still performed by susceptible agents getting in contact with an infected agent.

2.3.1 SEIRDV model equations

The SEIRDV model consists of five different states, which are schematised in figure2.2:

- S : Susceptible state, starting state for all agents population except for the first infected individual;
- E : Exposed state, in this state there are all agents that have been exposed to the disease; it models the incubation time agents are not infectious in this state;
- I : Infected state, in this state there are all agents that have developed the symptoms or are infective but asymptomatic and all are capable of infecting susceptible agents;
- R : Recovered state, in this state arrive all agents that have survived the disease or that may have been immunized. From this state it is not possible to be reinfected or loose the immunization;
- D : Dead state, in this state arrive all agents that unfortunately have not survived the disease;
- V : Vaccinated state, in this state arrive all agents that have been immunized

against the disease.

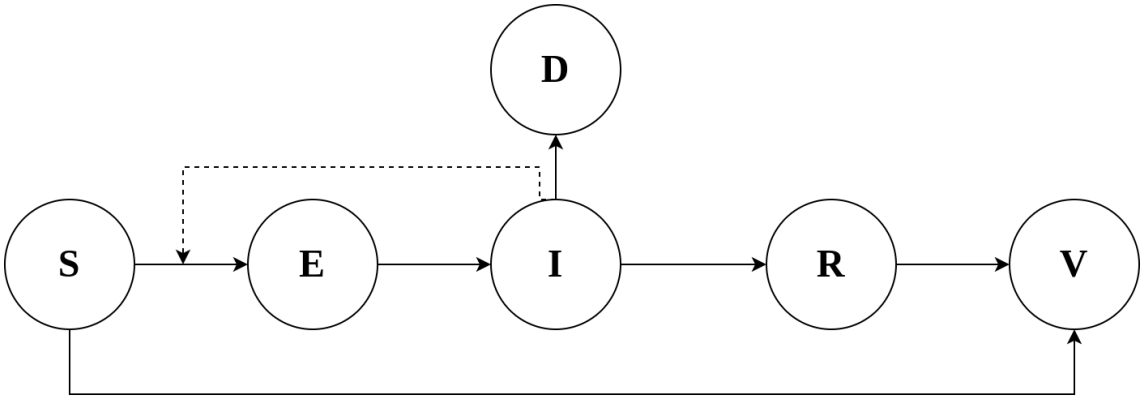


Figure 2.2: Transition scheme of the SEIRDV model

The total population at any time t is $N = S(t)+E(t)+I(t)+R(t)+D(t)+V(t)$ and the ODEs governing the system are:

$$\begin{cases} \dot{S} = -\beta \frac{I}{S+E+I+R+V} S - \delta_S S = -(\lambda + \delta_S) S \\ \dot{E} = \beta \frac{I}{S+E+I+R+V} S - \mu E \\ \dot{I} = \mu E - (\alpha + \gamma) I \\ \dot{R} = \alpha I - \delta_R R \\ \dot{D} = \gamma I \\ \dot{V} = \delta_S S + \delta_R R \end{cases} \tag{2.5}$$

where the model parameters are the same of those described in (2.1). Parameter μ is the transition rate from E to I. The force of infection in this case is:

$$\lambda = \frac{\beta I}{S + E + I + R + V} \tag{2.6}$$

2.3.2 Basic reproduction number

The computation of the effective reproduction number R_t associated to SEIRDV model follows a similar approach of the one described for the SIRDV model, with the main difference that now the infective compartments are E and I.

The equations for these compartments are: Defining:

$$\dot{E} = f_1(E, I) = \beta I \frac{S}{S + E + I + R + V} - \mu E$$
$$\dot{I} = f_2(E, I) = \mu E - (\alpha + \gamma) I$$

The linearization of the equations is

$$\frac{\delta}{\delta E} f_1(E, I) = -\beta I S \frac{1}{(S + E + I + R + V)^2} - \mu$$
$$\frac{\delta}{\delta I} f_1(E, I) = -\beta \frac{S}{S + E + I + R + V} - \beta I S \frac{1}{(S + E + I + R + V)^2}$$

$$\frac{\delta}{\delta E} f_2(E, I) = \mu$$
$$\frac{\delta}{\delta I} f_2(E, I) = -(\alpha + \gamma)$$

The resulting Jacobian matrix evaluated in the disease free equilibrium ($E = I = 0$) is:

$$\begin{pmatrix} \frac{\delta}{\delta E} f_1(0,0) & \frac{\delta}{\delta I} f_1(0,0) \\ \frac{\delta}{\delta E} f_2(0,0) & \frac{\delta}{\delta I} f_2(0,0) \end{pmatrix} = \begin{pmatrix} -\mu & \beta \frac{S}{S+R+V} \\ \mu & -\alpha - \gamma \end{pmatrix} = T + \Sigma$$

where

$$T = \begin{pmatrix} 0 & \beta \frac{S}{S+R+V} \\ 0 & 0 \end{pmatrix} \quad \Sigma = \begin{pmatrix} -\mu & 0 \\ \mu & -\alpha - \gamma \end{pmatrix}$$

The linearized ODEs at the DFE are:

$$\begin{pmatrix} \dot{E} \\ \dot{I} \end{pmatrix} = (T + \Sigma) \begin{pmatrix} E \\ I \end{pmatrix}$$

The NGM matrix is defined $K = -T\Sigma^{-1}$. Thus, first we compute Σ^{-1}:

$$\Sigma^{-1} = \begin{pmatrix} -\frac{1}{\mu} & 0 \\ -\frac{1}{\alpha+\gamma} & -\frac{1}{\alpha+\gamma} \end{pmatrix}$$

In fact:

$$\begin{aligned} \Sigma\Sigma^{-1} = & \begin{pmatrix} -\mu\frac{-1}{\mu} + 0 & -\mu 0 + 0 \\ \mu\frac{-1}{\mu} + (-\alpha-\gamma)\frac{1}{\alpha+\gamma} & \mu 0 + (-\alpha-\gamma)\frac{-1}{\alpha+\gamma} \end{pmatrix} \\ = & \begin{pmatrix} 1 & 0 \\ -1 + \frac{\alpha+\gamma}{\alpha+\gamma} & 0+1 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \end{aligned}$$

Finally K is:

$$K = -\begin{pmatrix} 0 & \beta\frac{S}{S+R+V} \\ 0 & 0 \end{pmatrix} \begin{pmatrix} -\frac{1}{\mu} & 0 \\ -\frac{1}{\alpha+\gamma} & -\frac{1}{\alpha+\gamma} \end{pmatrix}$$

$$= -\begin{pmatrix} -\frac{\beta}{\alpha+\gamma}\frac{S}{S+R+V} & -\frac{\beta}{\alpha+\gamma}\frac{S}{S+R+V} \\ 0 & 0 \end{pmatrix} = \begin{pmatrix} \frac{\beta}{\alpha+\gamma}\frac{S}{S+R+V} & \frac{\beta}{\alpha+\gamma}\frac{S}{S+R+V} \\ 0 & 0 \end{pmatrix}$$

Thus it is possible to conclude with:

$$R_t = \rho(K) = \frac{\beta}{\alpha+\gamma}\frac{S}{S+R+V}$$

2.4 BioPEPA formal language

BioPEPA is a formal language, extension of the PEPA language[32], for modelling biochemical networks. PEPA, acronym of Performance Evaluation Process Algebra, is a process algebra defined for modelling computer systems and obtain their performance analysis. BioPEPA extends it in order to handle some features of biochemical networks, such as stoichiometry and different kinds of kinetic laws. A key advantage of BioPEPA as compared to its predecessor PEPA is the higher level of abstraction that allows an easier writing of the models. Another feature of BioPEPA is the possibility to support different kinds of solvers, including stochastic simulation, analysis based on ordinary differential equations (ODEs) and model checking in PRISM[33]. The following section resumes the key concepts necessary to define within BioPEPA the epidemiological models used in this project.

For the full description and discussion of BioPEPA formalism please refer to its original paper[18].

2.4.1 BioPEPA syntax and capabilities

A BioPEPA model is composed of:

- model components (or compartments);
- functional rates;
- system equation.

The model compartments represent different states/sets in which an agent belongs, and each includes the list of actions that can be undertaken by the agents. The compartment definitions describe how an agent undertaking an action affects the compartment itself. The functional rates identify a possible action made by an agent inside a compartment. The rate value represents the speed at which an activity is performed or equivalently the mean fraction of time agents reside in the component. The system equations describe how each compartment is related to the others, namely in synchronisation or parallelization, thus the complete behaviour of the system.

BioPEPA syntax is defined as a combination of symbols to represent the biochemical species (the agents are single molecules) and their actions inside the system. The model can be defined following these rules, which are resumed in general concepts from their original biochemical meaning.

The definition of the three main components of BioPEPA are:

1. Functional rates: A functional rate, $f_{rate} \in \mathbb{R}$, is defined in BioPEPA as a constant number or derived by a function of possible parameters sk, and of the number of agents in a compartment C. The following scheme resumes the possible combinations on sk and C to define a rate (the symbol '—' is a separator of the possible definitions).

$$\begin{aligned} f_{rate} &::= f(sk, C) \mid f(sk) \mid sk \\ sk &::= int \mid float \mid name \mid sk + sk \mid sk - sk \mid sk * sk \mid sk/sk \mid sk^{sk} \mid \\ &\quad exp(sk) \mid log(sk) \mid sin(sk) \mid cos(sk) \end{aligned}$$

   where: C is the compartment value i.e. the cardinality of the set, f is a function returning a value belonging in the definition of sk.

2. Components: A component of the system is identified with an unique name and its definition which is assigned with the ' =' operator. The definition is composed of actions undertaken by agents with the assigned rate (α) or a composition of other components. The following scheme resumes the possi-

ble definitions of a generic component S ('—' is a separator of the possible definitions).

$$S ::= (\alpha, k)\text{op}S \mid S + S \mid C$$

where: α is a functional rate, k is the number of agents involved (default is 1), + is the choice operator between two activities, op is one of the following operations

- $\downarrow$ (reactant) reduce the agents in the component;
- $\uparrow$ (product) increase the agents in the component;
- $\oplus$ (activator) necessary element in the activity.

3. System equation: The system equation can be expressed as:

$$P ::= P \underset{L}{\bowtie} P \mid S(x)$$

where: $\bowtie$ is the cooperation symbol which synchronise the components, L is the set of activities of cooperation (symbol * means every shared activity), x is the initial state of the compartment S.

2.4.2 BioPEPA as epidemiological tool

BioPEPA can be adapted to perform epidemiological simulations as described in the paper [34] published by Federica Ciocchetta, one of the authors of BioPEPA. The advantages of using PEPA or BioPEPA to describe an epidemic can be found in the cross competency of the scientific fields of epidemiological studies: epidemiology requires both medical and biomedical expertise, as well as mathematical and statistical knowledge. In BioPEPA the syntax rules are simple and so that it can be exploited by biomedical scientists or other professionals who are not confident with the mathematics of modelling and simulations like computer scientists and mathematicians.

To describe a BioPEPA model no coding skills are required, and all the complex mathematical equations and matrices needed for the simulations are generated automatically by the BioPEPA tools. The main advantage of BioPEPA as compared to PEPA is the straightforward description of the population classes among different sets/compartments and their links. Each compartment represents both the status and the type of agent which resides inside. The compartment definition describes the activities and the mean time required to be performed by the agents, such as entering in contact between each other, developing symptoms or healing from the disease.

Basic SIR model in BioPEPA

The following BioPEPA model defines three compartments representing a classless SIR epidemiological model.

The agents in the S (susceptible) compartment can be infected and pass to the I (infected) compartment, from there it is possible to recover and pass to the R (recovered) compartment. The rates at which each agent performs an action are defined with the action itself. In this example there are two rates, α and β, that are constant during the simulation.

The system equations put all compartments in collaboration over the common actions. The initial state is omitted in this setup of the model.

Functional rates:

$$infect = \alpha, recover = \beta$$

Compartments definitions:

$$\begin{aligned} S &\stackrel{def}{=} (infect) \downarrow S \\ I &\stackrel{def}{=} (infect) \uparrow I + (recover) \downarrow I \\ R &\stackrel{def}{=} (recover) \uparrow R \end{aligned}$$

System equation:

$$S \underset{*}{\bowtie} I \underset{*}{\bowtie} R$$

Basic classed SIR model in BioPEPA

As in the previous example, in the following model the agents have three possible states (S,I,R). In addition this model divides the population in two classes, namely A and B, governed by different rates (for example to describe different risk classes). To represent a classed SIR epidemiological model six compartments are needed, i.e. the Cartesian product of states and classes. Since now there can be two types of interactions leading to infection, in the model definition it is now necessary to represent the interaction between a susceptible agent in one class and an infected agent belonging to the other class. This can be done via the addition of the first class infecting activity in the infected compartment of the second class as activator (functional rate called $infectAB$), and vice versa (functional rate $infectBA$). In fact, this is equal to the chemical catalyst concept represented in BioPEPA with the $\oplus$ symbol.

The functional rates in this case are:

$$infectAA = \alpha, \quad infectAB = \alpha_1, \quad recoverA = \beta$$
$$infectBB = \gamma, \quad infectBA = \gamma_1, \quad recoverB = \delta$$

Compartments and their actions are then defined as:

$$\begin{aligned}
S_A &\stackrel{def}{=} (infectAA) \downarrow S_A + (infectAB) \downarrow S_A \\
S_B &\stackrel{def}{=} (infectBB) \downarrow S_B + (infectBA) \downarrow S_B \\
I_A &\stackrel{def}{=} (infectAA) \uparrow I_A + (infectAB) \uparrow I_A + (infectBA) \oplus +(recoverA) \downarrow I_A \\
I_B &\stackrel{def}{=} (infectBB) \uparrow I_B + (infectBA) \uparrow I_B + (infectAB) \oplus +(recoverB) \downarrow I_B \\
R_A &\stackrel{def}{=} (recoverA) \uparrow R_A \\
R_B &\stackrel{def}{=} (recoverB) \uparrow R_B
\end{aligned}$$

Finally, the system equation is:

$$S_A \underset{*}{\bowtie} S_B \underset{*}{\bowtie} I_A \underset{*}{\bowtie} I_B \underset{*}{\bowtie} R_A \underset{*}{\bowtie} R_B$$

BioPEPA models files

The two SIR models just described are represented in `.biopepa` files as follows. SIRmodel.biopepa

```
1 infect = [alpha];
2 recover = [beta];
3
4 S = infect<<S;
5 I = infect>>I + recover<<I;
6 R = recover>>R;
7
8 S <> I <> R
```

Listing 2.1: Basic SIR model in BioPEPA

ClassedSIRmodel.biopepa

```
 1 infectAA = [alpha];
 2 infectAB = [alpha_1];
 3 recoverA = [beta];
 4 infectBB = [gamma];
 5 infectBA = [gamma_1];
 6 recoverB = [delta];
 7
 8 S_A = infectAA<<S_A + infectAB<<S_A;
 9 S_B = infectBB<<S_B + infectBA<<S_B;
10 I_A = infectAA>>I_A + infectAB>>I_A + infectBA(+) + recoverA<<I_A;
11 I_B = infectBB>>I_B + infectBA>>I_B + infectAB(+) + recoverB<<I_B;
```

```
12 R_A = recoverA>>R_A;
13 R_B = recoverB>>R_B;
14
15 S_A <> S_B <> I_A <> I_B <> R_A <> R_B
```

Listing 2.2: Basic classed SIR model in BioPEPA

Chapter 3

Workbench and extensions

The BioPEPA project provides two different tools to simulate a BioPEPA model with different algorithms and techniques. They are the BioPEPA Workbench and the BioPEPA Eclipse Plugin.

The Workbench is a command line tool without graphical interface which provides many features, but, unfortunately, it is scarcely documented and its latest release is from 2008. The Eclipse Plugin, instead, is an extension to the well known Eclipse IDE[35], thus providing a graphical interface together with the IDE editor and its latest release is from 2014. In the beginning of the project, the Eclipse Plugin seemed to be the best and most updated option, and thus it has been chosen to run the simulations for the models of this project. However soon two fundamentals problems made its use impossible. A realistic simulation of an epidemic requires the possibility of updating the model variables or providing progressive data with the simulation time progression (e.g., to take into account changes in transmission).

The Eclipse Plugin has the possibility to introduce a time variable, but only in simple mathematical expressions for the definition of the activity rates. The Eclipse Plugin has not the possibility of defining code functions to encapsulate any complex set of operations. Furthermore, the Eclipse Plugin does not allow automatic iterations and simulation launches. These limitations of the Eclipse Plugin can be overcome by using the BioPEPA Workbench. The Workbench is a more flexible tool and open to integrations, since there is a direct access to the source files of the simulation which are compiled for every run. Since it is a command line tool, scripts can automate the iteration of solvers over different models and/or parameters. Thus, the Workbench is a much more suitable tool for programmers and researchers. Also the workbench had its disadvantages as they will be discussed in this chapter.

The BioPEPA Workbench, available at the official website[36], features and orga-

nization is presented in the next section along with modifications and integrations proposed. The BioPEPA Workbench utilises many programs and libraries which have their own requirements to work properly (for details see the requirements specification at the online repository [37]). The additional tool designed in this thesis to generate a large BioPEPA model automatically is described in the second part of the chapter.

3.1 BioPEPA Workbench integrations

3.1.1 Workbench structure

The BioPEPA Workbench, in the following indicated with BPWB, has different subroutines which are managed by a global shell script. This tool is made of different components, library (main simulator solver is StochKit) and takes advantage of other programs, all elements which are not fully described in the Workbench manual. The following tree resumes the folders structure [1]:

```
bpwb
    bin:  the workbench executables
    cpp:  model translations in cpp code for stochkit
    css:  css file for the html report page
    dat:  output results of the simulations
    dizzy:  model translations in dizzy format
    doc:  documentation and manual of the workbench
    dot:  files for dot representation of the states
    eps:  files for producing the pdf report
    gnu:  gnuplot command files to plot the results
    highslide:  javascript files for the html report page
    js:  javascript files for the html report page
    log:  logs of executions
    matlab:  model translations in matlab
    pdf:  pdf reports
    png:  graphs produced by gnuplot
    prism:  model translation in prism format
    src:  ml source files of the workbench
    stochkit:  simulator files for stochkit
    sundials:  simulator files for sundials
    tex:  latex report files
    thumbnails:  graphs of gnuplot for the html page
    vfgen:  vf files for model translation in sundials source code
stochkit
    test
```

[1] Files and StochKit library folders omitted

```
└── BIOPEPA: model files are moved here to run the simulations with
    stochkit
```

The BPWB global script is called **bp** and it launches the workbench on every **.biopepa** files along with the pairing **.csv** files that are found inside the **bpwb** folder. The program produces the simulator files of **stochkit**, as well for **matlab** and the model representations in **.vf** format for the library converter **VFgen** and the model translation in **dizzy** and **prism**. The workbench also prepares reporting files like the model description in Latex documents. All these files are moved in the belonging folders by the **bp** script. The script proceeds to compile and run the simulator processes. Eventually, it moves the result files in the appropriate **dat** folder and calls some post-processing programs like **gnuplot** to generate the simulation graphs.

The **bp** script instructions control flow is based on the configuration parameters defined in the **biopepa.cfg** file. It contains also input configurations for all the programs called by the global script, the most important is the **simulation.stoptime** which is a final time for the duration of the time inside the simulation. The simulation actual final time then can only be lower or equal to the configuration stoptime. In case a simulation stops before the limit, it means that no activities could be performed by any agent in any compartment, e.g., when in a SIRD model no agent is infected and thus there cannot be new infected or recover or deaths, and all agents are in S, R and D compartments.

3.1.2 Workbench and StochKit integrations

In order to elaborate the epidemiological models, it is required to make available the data required for each rate at different time values. This is a requirement to include the R_t coefficient. This necessity is resolved with the creation of custom functions which in turn require the value of the variable **time**. The custom functions are called by the rate definitions inside the model thus to these definition it must be made available the variable **time**.

Time variable recognition

A simple test of the workbench to determine the correct usage of the **time** variable highlighted that code parsing section required some modifications to recognise the **time** variable and to include the desired features.

The parser in the workbench analysed already all the tokens necessary to describe rates. In fact inside the data structures of specific recognised tokens, the variable **time** was already present inside the **VFgen** module but not inside the

StochKit one. This problem is here overcome by adding the variable **time** in the special token list for the workbench **StochKit** simulator module. Thanks to this modification the libraries **StochKit** and **Sundials** are suitable as the stochastic simulator and the as ODEs simulator respectively.

StochKit updates

The original **StochKit** module as available together with the Workbench[19], does not recognise the **time** variable. This is due to the data structures and the function containing the model description do not have access to the **time** variable in the simulator code. The **time** variable is a local variable in the loop of the main function of the simulator. In order to use it inside the rates of the model description created by the workbench, it is necessary to add it to the definitions and declarations of the functions in the **StochKit** library.

For what concern the loading of model parameters that change in time, such as the R_t coefficient, and including them inside the definition of the rates, this can be easily done in the workbench thanks to the possibility of calling external functions in C++ language. In order to define the temporal-changing coefficient in the rates it was required to define a custom function that provides the value inside the **KineticFunctions.cpp** file, as described in the BPWB manual.
Additional function for reading indexed values $n \in \mathbb{R}$ from a CSV file can be found in the appendix A.

Sundials updates

The Sundials simulator computes the numerical solution to the ODEs associated to the BioPEPA epidemiological model. The code for the **Sundials** simulator is generated by the BPWB through the **VFgen** tool. As already mentioned, the **time** token is correctly recognised for the generation of the input file for VFgen, thus the original version of Sundials already handles the time value in the model definitions.

Unfortunately, while testing the parsing and call of a loading function for the R_t value inside the **Sundials** simulator, generates an additional problem. As a matter of fact, the code for simulating the model with Sundials is generated by **VFgen**, which is based on **GiNaC** library, which in turn uses the **CLN** library. **VFgen** supports the definition of custom functions, but only if they are mathematical expressions. Using a C call directly like in the case of **StochKit** is not possible. Modifying **VFgen** to support a direct function call as described would necessarily mean the modification of **GiNaC** too. The use of an ODE solver that could read an external parameter is then possible only undertaking one of the following actions:

- Modyfing **VFgen** and **GiNaC**;
- Try to use the **Matlab** solver for ODEs, which files are already generated by **VFgen**; however this suite is not free and open source;
- Try to use the **Dizzy**[38] simulator, which provides both stochastic and ODE solvers. Dizzy is based on another kind of formalism and translating a BioPEPA model into a Dizzy model before running simulations seems like a waste of time. Furthermore, the Dizzy formalism and modelling tool are poorly documented, even less than BPWB;

Considering the available options, none of them are worth the time needed to implement the many changes necessary. So the decision is to work around the Sundials problem: we propose to use a placeholder variable in the BioPEPA model. After the workbench has generated the simulators code, the placeholder variable is substituted with a clever "find and replace" script inside the C++ and C code of **StochKit** and **Sundials** respectively before the simulation run. Since BPWB do not provide a dedicated file for custom functions for Sundials, the file **CustomFunctions.c** was created inside the **sundials** folder.

"Find and replace" script

The find and replace script is written in Python, which is a language particularly suitable to handle word processing. The script enhances the simple "find and replace" logic by exploiting the pattern that BPWB uses to declare variables inside the StochKit and VFgen generated files. Using a configuration file in input, the script finds the simulators code inside the corresponding folders and performs the substitution of all placeholder variables.

This modification of the code is not a clean solution, but it was safer, than other options, since modifying huge libraries like GiNaC and CLN would probably introduce bugs or an unexpected behaviour, as well as requiring a huge amount of time just to include a function call. Furthermore, it is quite easy to implement the substitutions of placeholders, and this considerably expands the flexibility and capabilities of the simulators for BioPEPA models.

3.1.3 Workbench usage

Some condition must be satisfied in order to launch the Workbench on a BioPEPA model file:

- set global simulation configurations in the **biopepa.cfg** file;
- the **modelname.biopepa** file containing the BioPEPA model definition must

be in the **bpwb** folder;

- the **modelname.csv** file containing the model starting state values for the rate constants and compartments population;
- if the model contains calls to custom functions, the functions specification for each placeholder rate must be in the JSON format with the name **custom_functions_configMODELNAME.json**;
- any file that is read or called by custom functions must have a copy present in the **bpwb** and **sundials** folders (unless the custom functions handle the file path or any other loading process).

When all conditions are met, it is possible to call the **./bp** script in a terminal from the **bpwb** folder. The script will present text output to inform on the execution status. The data files containing results of the simulators computations are placed in the **bpwb/dat** folder. The graphical representation of the simulation results generated by **gnuplot** can be found in the **bpwb/png** folder. Other intermediate computation files plus the model conversion for PRISM and Dizzy formalism are organized in appropriate subfolders of **bpwb**.

The files to generate the simulation of section 4.1.1 as file format examples are reported in the appendix A.

3.2 Epidemiological model script

The definition in BioPEPA of the epidemiological model with classes, as described in chapter 2.4.2, requires the setup of a large number of interactions, activities and rates, which scale quadratically with the number of states. This process cannot be performed manually when there are several classes, mainly for the risk of introducing errors. Thus, there is the need for an automatic tool that construct the definitions of the model starting from a table of the model rates among states. In addition to the BioPEPA Workbench integrations, this thesis work provides in the new repository of the code[37] a model writer program. The program takes two files as input: the model description and details, class contacts matrix and the name of the model as described in section3.2.2. The program computes the BioPEPA file containing the model and the CSV file with the initial status and the values of the other variables which then are written in output.

This program is combined with the "find and replace" script. It has an additional file in output, which is the configuration file for the placeholders put in the model and it can be directly used with the script.

The model writer is a OOP[2] program written in Python, as it is a language well suitable for text elaborations, as already mentioned. The application is conceived to be flexible, but most importantly expandable for anyone that desires to upgrade or define new models.

3.2.1 Model writer structure

This section describes the structure and main duties of each class.The following tree represents the structure of the source directory:

```
└── src
    ├── modelImplementation.py
    ├── models
    │   ├── SIRDModel.py
    │   └── SIRDvModel.py
    ├── model-writer.py
    ├── substituionsInfo.py
    └── transition.py
```

The model writer program logic consists of two parts: general model code and specific model code. These are implemented in two corresponding files, the **model-writer.py** contains the main function of the program and handles the general logic. In particular, it carries out input checks and handles the output logic.

The second file is variable, its name is given in input and the **model-writer.py** loads the module to execute the specified model logic. This file must contain an object definition that respects the model implementation logic and may hold custom data for this purpose. This class must have public the methods defined in the **modelImplementation.py** abstract class, which functions also as an interface. These methods are called by the general algorithm and they have the following responsibilities:

1. **computeTransmissionRates**: this method is the first to be called by **model-writer.py** and it has the responsibility of computing and defining all activities of the model;

2. **computeModelDefinitions**: this method is the second called by the general logic and it has to return all the model components definitions;

3. **computeSystemEquation**: this is the last method called for the model computation and it computes the system equation of the model.

4. **getSubstitutions**: this method is called at the end of the computation and returns the placeholder substitutions, which will be used to generate the con-

[2]Object Oriented Programming

figuration file for the substitution script;

3.2.2 Model writer usage

The model-writer program is a command line tool that has to be launched with the following arguments:

- flag -m: path to the Python file containing the implementation of the model to be constructed;
- flag -c: name of the class implementing the methods described in 3.2.1;
- flag -p: path to the JSON file containing the configuration for the model to be constructed;
- flag -i: path to th CSV file containing the contact matrix between the model classes;
- flag -o: name for the biopepa file in output.

The files to generate the simulation of section 4.4 as file format examples are reported in the appendix A.

Chapter 4

Simulation results

This chapter presents the application of the epidemiological models described in chapter 2 to simulate and analyse a realistic epidemic. As a first example of application, we consider the simulation of the Italian Covid-19 epidemic using both the SIRDV and SEIRDV models. As a second example, we present the results of classed model which is easily extendable with more classes and real data. The formula to include the R_t factor in the models is derived from *Gatto et al.*[14].

4.1 Simulation of the Italian COVID-19 pandemic

4.1.1 Results of the SIRDV model

The SIRDV model simulation of the Italian Covid-19 epidemic takes into account the epidemiological data available at [39]. In particular, the model results are based on the R_t values estimated by the ISS [40]. In fact, the functional rates governing the transmission are changed in time in order to match the modelled R_t with the real estimation from the data (as described in Section 2.2), the R_t trend is depicted in figure 4.2.

The simulation spans a time frame of 604 days, from 06/02/2020 to 30/09/2021. This period covers the first three waves in Italy caused by the first strain of SARS-CoV-2 to similar variants (widely known as Alpha, Beta, Gamma and Delta variants) and the first 10 months of the vaccination campaign. The Omicron variant, that had strong differences in the virus transmission, became relevant in Italy after the end the simulation period consider in this thesis.

The results are compared against the recorded active cases, the total recovered and the total dead cases. From the simulation results, thanks to the additional $CUMI$ compartment and the D compartment, the daily infections and deaths are

inferred and compared with the recorded national data.

The model starts on the 06/02/2020 with two infected individuals. This initial date, together with the imposition of the R_t, allows the model to well reproduce the main features of the epidemic curve. No calibration efforts on the initial conditions are made to obtain perfect matching with the historical data of the healthcare authorities. The calibration of the model parameter is a sub-problem that is out of the scope of this project.

The model SIRDV simplifies the vaccination process by removing the vaccinated individuals from the transmission dynamics (as if the vaccine would provide full immunisation). The simulation then ignores the distribution of the first doses, and imposes that the number of vaccinated individuals among S and R compartments matches the daily number of second doses distributed in Italy. Booster doses are excluded in this model because their distribution started after the time frame considered. The BioPEPA formalization of the model can be found in the appendix A.

Figure 4.1 shows the schematic representation of the as prepared by the file output `SIRDV.png` of the workbench. The same model is then automatically simulated

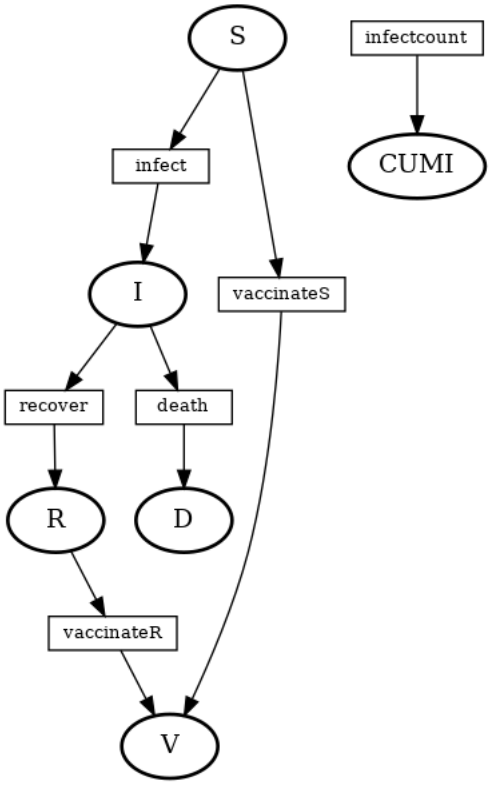


Figure 4.1: Transition scheme of SIRDV BioPEPA model

with ODE and stochastic Gillespie solvers.

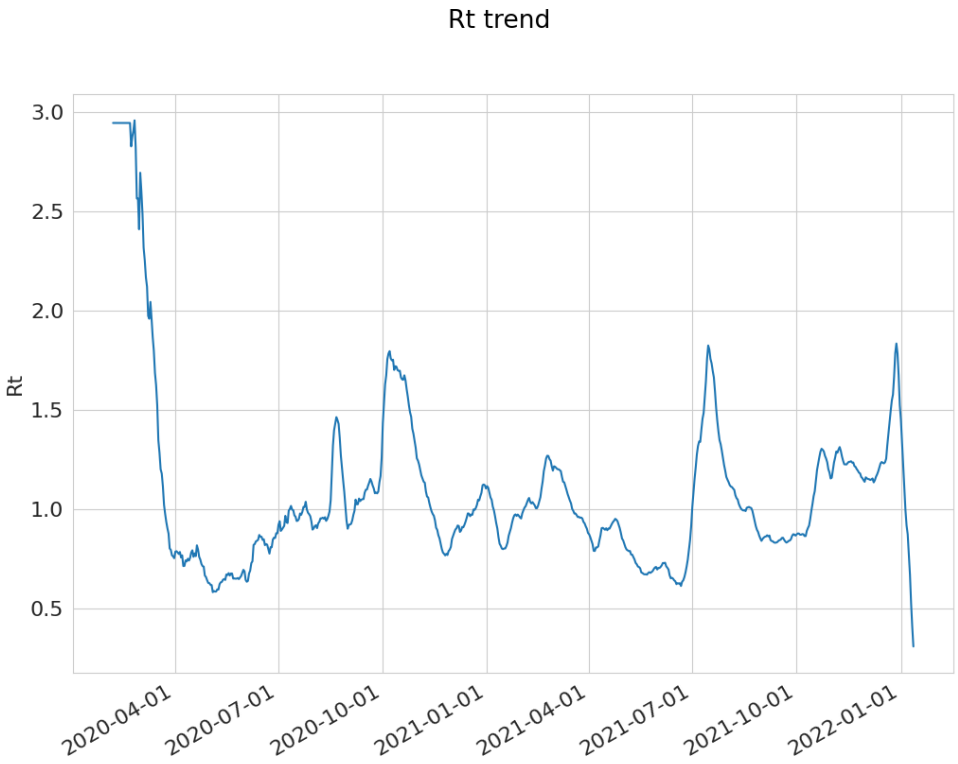


Figure 4.2: R_t coefficient trend in the simulations time frame, loaded by $f_{R_t}(t)$

Model parameters

The table 4.1 reports the values of the parameters and initial state of the simulation. In particular γ and α are the healing rate and death rate respectively. Their sum (0.2 days^{-1}) corresponds to the inverse of mean time spent by an individual in the I compartment, here imposed equal to 5 days. The death rate $\alpha = 0.004$ days^{-1} implies that the expected number of deaths in a long-term simulation is 2% of the infections. The `placeholder` and `vaccinerate` key labels are the placeholders used to substitute the appropriate function calls for the time dependant values of R_t and vaccine doses distributed in a day. In the simulation each time unit represents a day, thus all rate values are expressed accordingly. The functions $f_{R_t}(t)$ and $f_v(t)$ retrieve the values of the day depending on t rounded to the nearest integer.

Simulation results

Figure 4.3 presents a comparison among the epidemiological data and the simulation results in the different compartments, for the two considered solver. It is clear that the model results have a similar dynamic to the data, reproducing the timing and the order of magnitude of the epidemic waves in the active cases (I). This (not obvious) result is a direct consequence of the strategy adopted to perform the simulation, i.e.

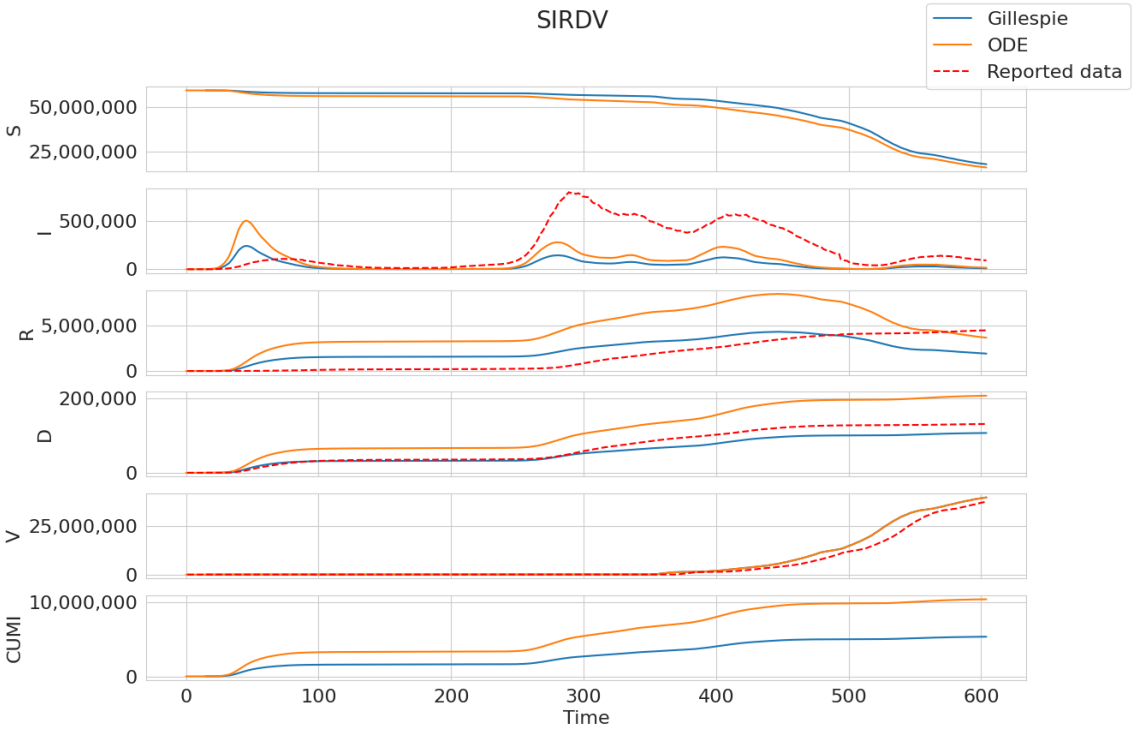


Figure 4.3: SIRDV simulation results: number of individuals in each compartment of the model as computed by the stochastic Gillespie solver (blue line) and by the ODE solver (orange line). The dashed red line show the corresponding epidemiological data.

Parameter	Value
placeholder	$f_{R_t}(t)$
vaccinerate	$f_v(t)$
γ	0.196 days^{-1}
α	0.004 days^{-1}

Compartment	Initial state
S	59641488
I	2
R	0
D	0
V	0
CUMI	2

Table 4.1: SIRDV Parameters values and initial state definition

the imposed equality between the modelled R_t and the one estimated from the data.

Since a proper calibration is missing, the data are represented in the figure just to give an idea of the real dynamics, and it was foreseen that the result would not exactly match the records. Nevertheless, the epidemic behaviour is well represented and consistent with the data.

There are several reasons why, the model results do not match this data. First of all, to obtain a number of deaths similar to the one recorded during the first wave, the model needs a much larger number of infected. This could be an indicator of the lack of a proper testing capacity during the first months of the pandemic. Furthermore, it must be taken into account that the active cases in the simulations depend only on the healing rate (about 5 days, meanwhile in the bulletins reports of healthcare authorities an active case is considered recovered only after a negative test. This implies that in some periods the number of infected individuals could have been overestimated. However in the first wave the contrary happens: the testing capabilities in the first months of the pandemic were unable to match the demand with orders of magnitude of difference.

Compartments R and D also show a good trend in comparison with the data. The decreasing of recovered cases, in the last 100 days of the simulation, is due to the fact that also recovered individuals have been vaccinated, thus in the SIRDV model they are moved from R to V.

The vaccinated compartment show a complete matching between the ODE and stochastic solvers results and a very close match with the historical data, which are here used as an input to the model.

As stated in the previous paragraph, the results are not completely comparable between compartments values, in particular for I (active cases). Comparing the daily cases of infections and deaths provides a better idea of the differences among the simulation results and the data, as reported in figure 4.4. Both daily trends are reproduced with high reliability by the solvers, but there are some important differences among Gillespie and the ODE solver. Even if the model parameters are

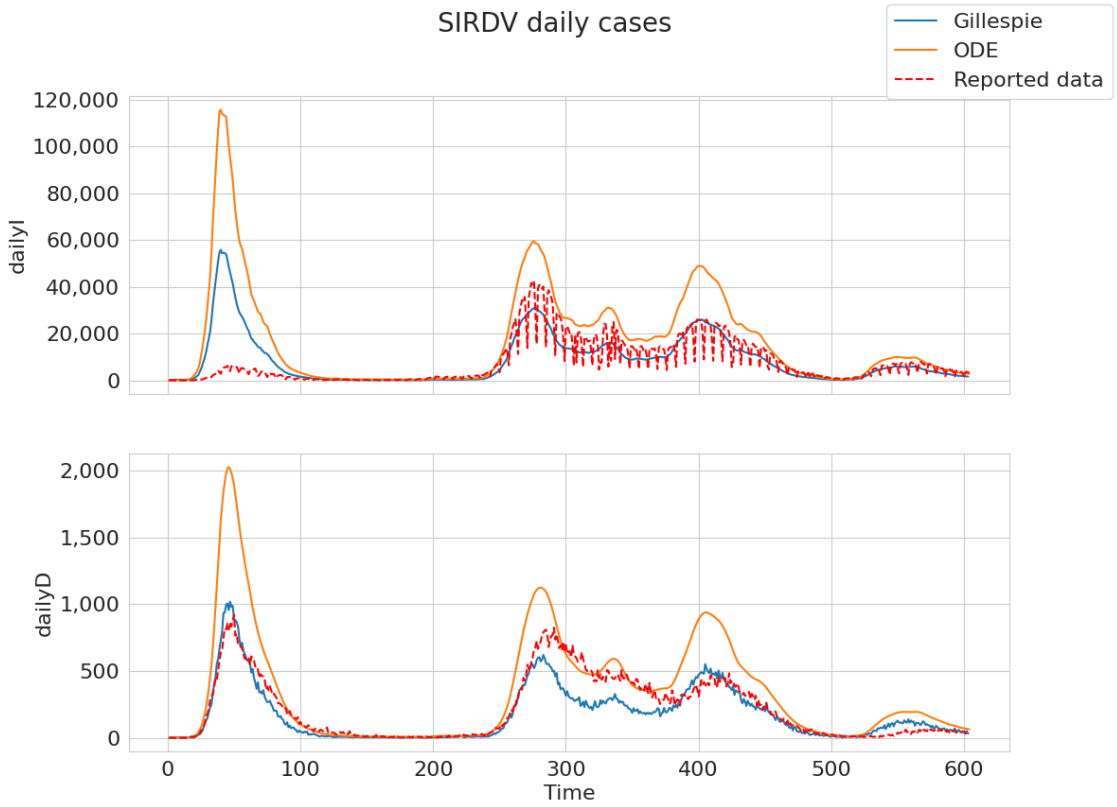


Figure 4.4: SIRDV daily infected and deaths: comparison among the reported daily infected and deaths (dashed red line) and those computed in the two solvers, Gillespie (blue line) and ODE (orange line).

the same, the stochastic solver provides a lower peak in all the waves

4.2 SIRDV model with vaccination impact

An interesting analysis that is typically performed with this kind on epidemiological models is the evaluation of the impact of the vaccination campaign. This section presents two simulation scenarios that replicate the same transmission rates evaluated under the assumption of the full vaccination (computed from the described run of the SIRDV model), but deploying lower numbers of daily vaccination doses.

The two scenarios are performed on a SIRDV model with only a fraction of the total doses administered (50% and 66%, respectively named SIRDV_vimpact05 and SIRDV_vimpact066).

Besides the different vaccination, this simulations are a replica of the one described in the previous section. The simulations span a time frame of 604 days, from 06/02/2020 to 30/09/2021. The model is again the SIRDV, thus a single vaccination is considered able to fully immunize an individual. The model results are compared in figure 4.9 against the recorded daily active cases, the total recovered and the total dead cases and against the results of the SIRDV simulation with the full vaccination doses. The outputs include the daily infections and deaths, wchich are compared with recorded data and SIRDV simulation. The BioPEPA formalization of the model can be found in the appendix A.

Simulation parameters

Table 4.2,4.3reports the values for the parameters and initial state of the simulations. The difference between the two simulations is only on the parameter in charge of influencing the vaccination rate: `percentage` is a coefficient representing the fraction of doses actually administered. As in the previous simulation γ and α are the healing rate and death rate respectively. The `placeholder` and `vaccinerate` key labels are the placeholders used to substitute the appropriate function calls.

Differently from the first SIRDV simulation, the label `placeholder` is substituted with the function $f_\beta(t)$ which is responsible to load the corresponding value of β at time t. The coefficient β is the same of the equations 2.1 and it is loaded from a file compiled from the output of the SIRDV simulation. The file contains the values of β computed daily in the previous simulation (under the assumption of full vaccination regime). The inclusion of the β coefficient, instead of R_t, allows to reproduce the same amount of daily contacts of the previous simulation. Since now a lower rate of vaccination is imposed, there are more susceptible individuals and thus the risk of infection increases. The placeholder `vaccinerate` is substituted with $f_v(t)$, the same function of the previous simulation in section 4.1.1.

Parameter	Value
placeholder	$f_\beta(t)$
vaccinerate	$f_v(t)$
percentage	0.50
γ	0.196
α	0.004

Compartment	Initial state
S	59641488
I	2
R	0
D	0
V	0
CUMI	2

Table 4.2: SIRDV_vimpact05 Parameters values and initial state definition

Parameter	Value
placeholder	$f_\beta(t)$
vaccinerate	$f_v(t)$
percentage	0.66
γ	0.196
α	0.004

Compartment	Initial state
S	59641488
I	2
R	0
D	0
V	0
CUMI	2

Table 4.3: SIRDV_vimpact066 Parameters values and initial state definition

Results on the vaccination impact

From figures 4.5,4.6,4.7,4.8,4.9 it is possible to note that the reduced number of doses administered would have impacted severely the epidemic. In Summer 2021 the new infections would have a rampant increase and active cases would probably have an magnitude order to overwhelm hospitals. It is clear that vaccinations in this model have a determinant impact as for the assumption of complete efficacy in preventing the disease infection. However have to be underlined that this scenario is definitely unlikely as the government would certainly impose new restrictions or even a new lockdown to limit contagions.

Nonetheless the results of these simulations are useful to have a quantitative estimation of avoided infections thanks to the vaccination campaign. In particular in the model SIRDV_vimpact05 Gillespie results: 21.659.347 total avoided cases; meanwhile in the ODE results: 29.833.476 total avoided cases. In the model SIRDV_vimpact066 Gillespie results: 8.275.347 total avoided cases; meanwhile in the ODE results: 21.374.721 total avoided cases.

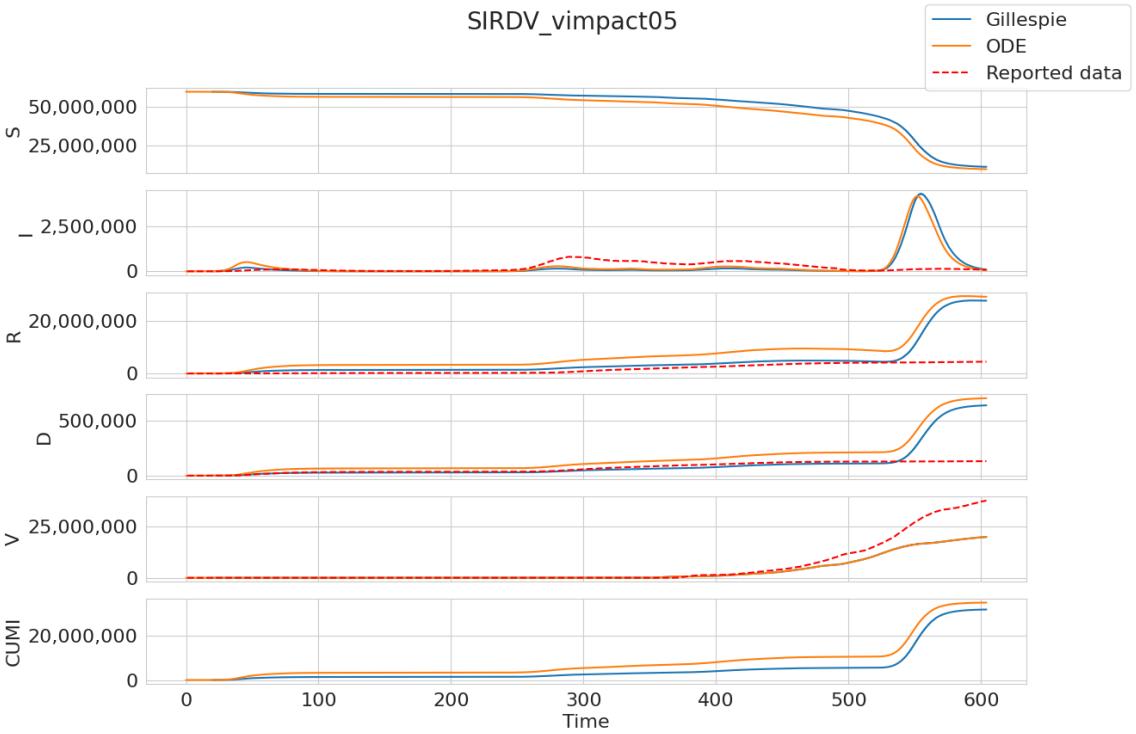


Figure 4.5: Same representation of figure 4.3 but for a 50% of the vaccination rate (scenario SIRDV_vimpact050)

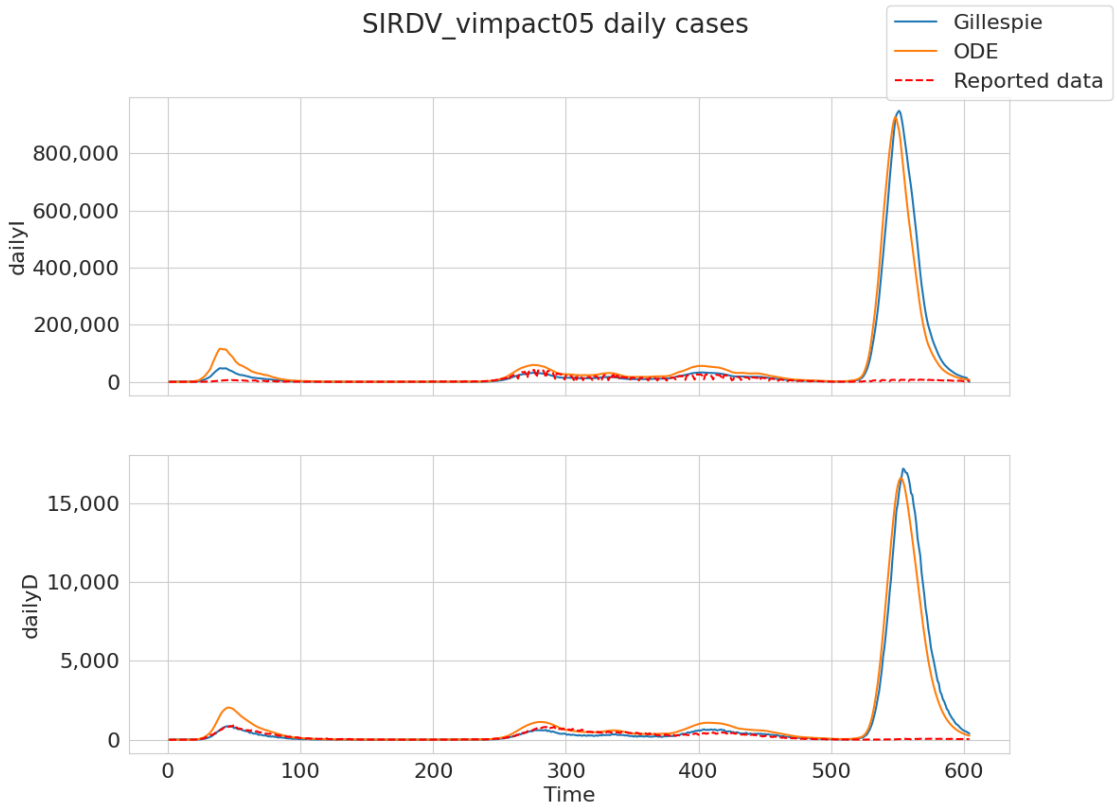


Figure 4.6: Same representation of figure 4.4 but for the 50% of the vaccination rate (scenario SIRDV_vimpact050)

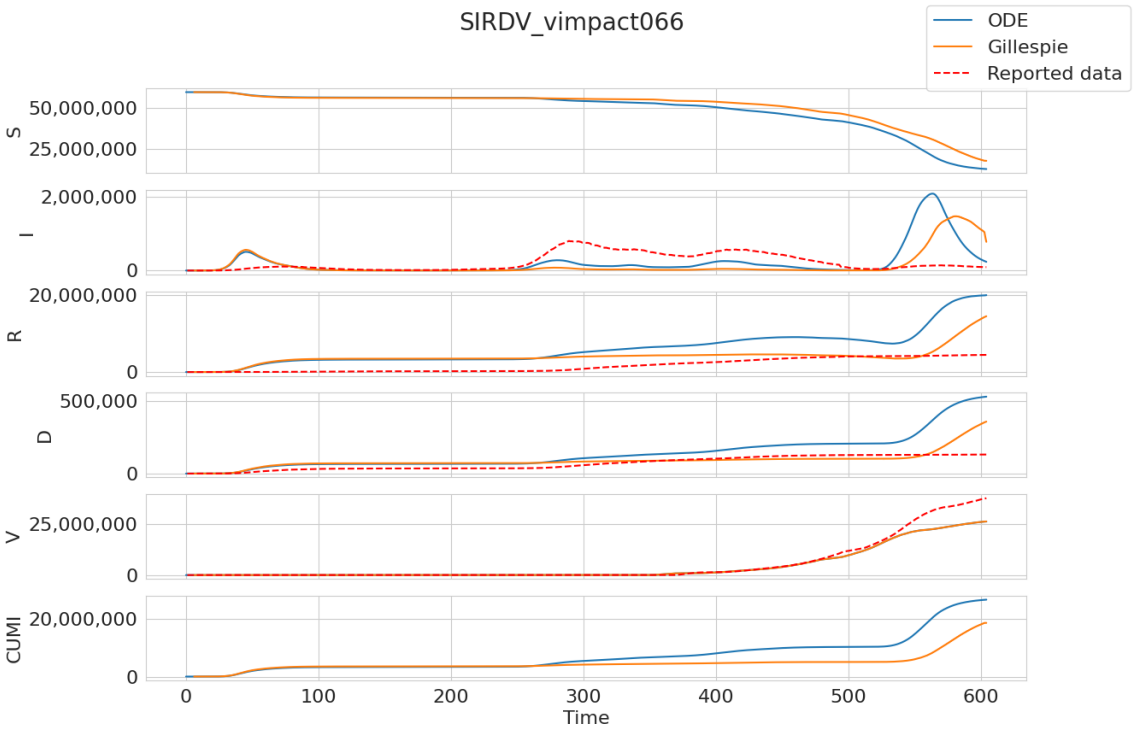


Figure 4.7: Same representation of figure 4.3 but for a 66% of the vaccination rate (scenario SIRDV_vimpact066)

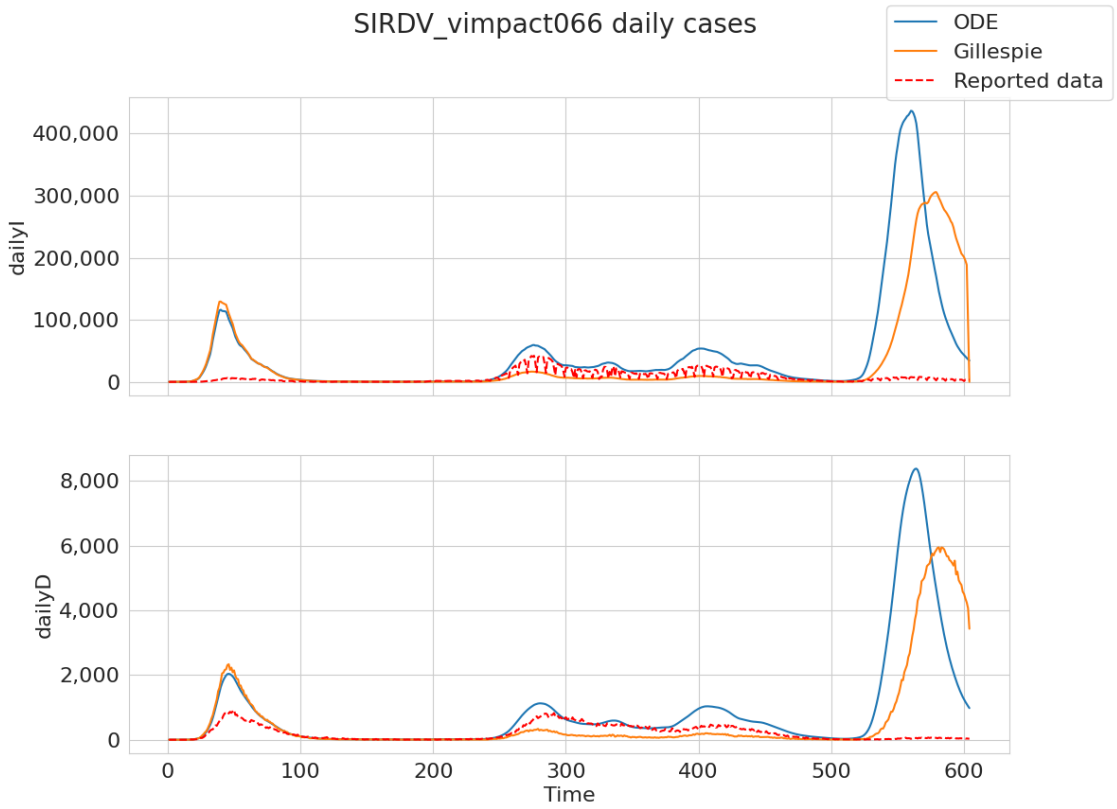


Figure 4.8: Same representation of figure 4.4 but for the 66% of the vaccination rate (scenario SIRDV_vimpact066)

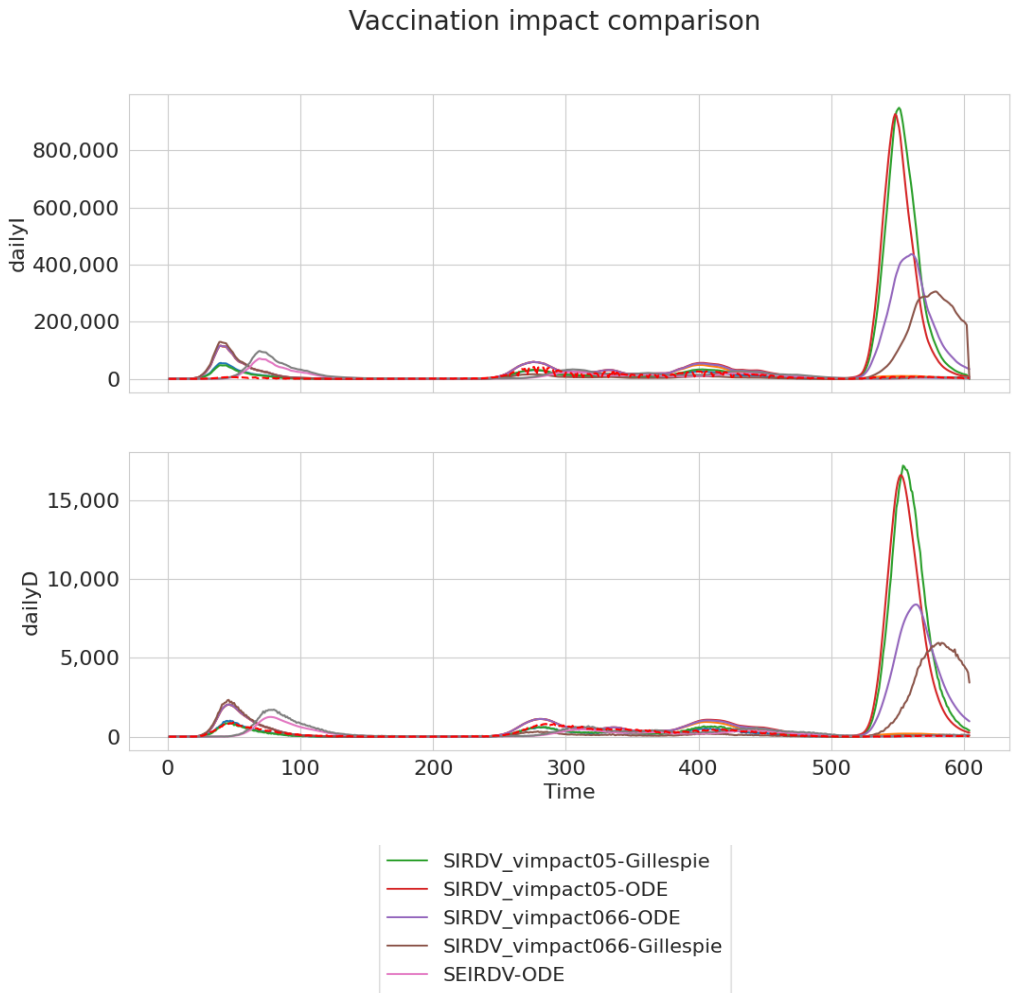


Figure 4.9: Vaccination impact comparison on daily cases and deaths of all SIRDV simulations

4.3 SEIRDV model

This section presents the results of the same analysis of the COVID-19 Italian epidemic performed with the SEIRDV model. The SEIRDV model is a more suitable model to simulate the COVID-19 transmission since it takes into account the incubation period of the disease (here assumed to be 2 days). Simulation results are compared with the recorded daily active, the total recovered and the total dead cases; also daily infections and deaths are inferred and compared as in simulation 4.1.1 but on a different period.

This simulations starts on the 7/01/2020 with 5 exposed individuals, proceeds for 604 days period, ending on 02/09/2021. The SEIRDV model requires an earlier date for the beginning of the epidemic than the SIRDV model, to give the exposed compartment E time to generate a congruent initial condition at 06/02/2020 for the I compartment. Besides for the E compartment, he same vaccinations assumptions and rates of the SEIRD model. As previous simulations, no calibration efforts on the initial conditions are made to obtain perfect matching with the historical data of the healthcare authorities. The BioPEPA formalization of the model can be found in the appendix A.

Figure 4.10 shows the schematic representation of the as prepared by the file output `SEIRDV.png` of the workbench.

Simulation parameters

Table 4.4 reports the values for the parameters and initial states of the simulation. In particular: γ and α are the healing rate and death rate respectively, `placeholder` and `vaccinerate` key labels are the placeholders used to substitute the appropriate function calls for the time dependant values of R_t and vaccine doses. The additional parameter δ represents the incubation period of 2 days. The functions $f_{R_t}(t)$ and $f_v(t)$ retrieve the values of R_t and the vaccination distributed doses in a day accordingly to the time variable t.

Parameter	Value
placeholder	$f_{R_t}(t)$
vaccinerate	$f_v(t)$
δ	0.5
γ	0.196
α	0.004

Compartment	Initial state
S	59641488
E	5
I	0
R	0
D	0
V	0
CUMI	0

Table 4.4: SEIRDV Parameters values and initial state definition

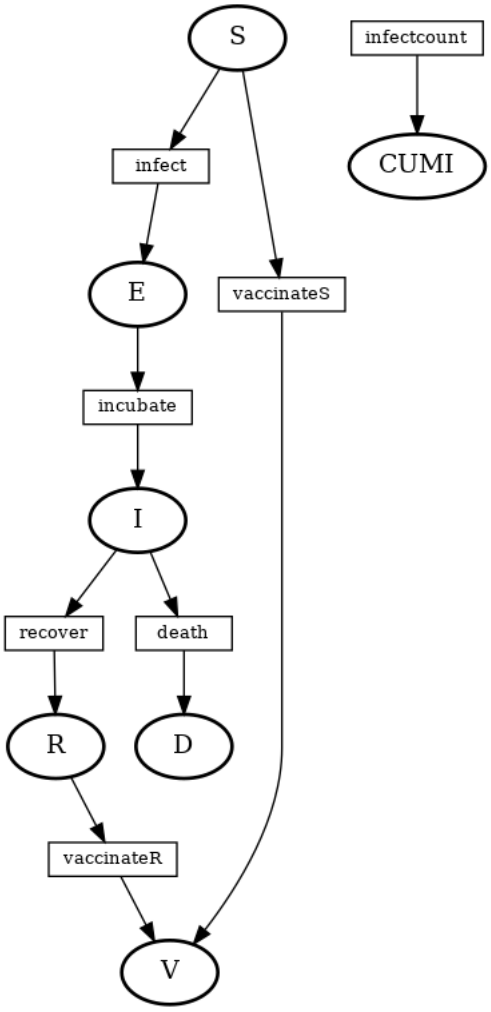


Figure 4.10: Transition scheme of SEIRDV BioPEPA model

Simulation results

The simulations results of the SERIDV model are interesting because as it is possible to see in figures 4.11,4.12, the active cases trend is quite different from the SIRDV simulations 4.1.1. In particular there is a better agreement between the simulators results, as the closer lines make evident, in respect to the SIRDV model results. This is probably due to the E compartment that act as a buffer since the exposed individuals are not capable of infecting other susceptibles. Furthermore in the second picture 4.12 the stochastic simulation results, as for the 4.4 figure, are much more coherent with the recorded data suggesting that the Gillespie algorithm could give better predictions if the model is calibrated.

4.4 Three class SIRD model example

The last example of application of the BioPEPA formal language consists in a classed SIRD model created with the Epidemiological model writer program described in

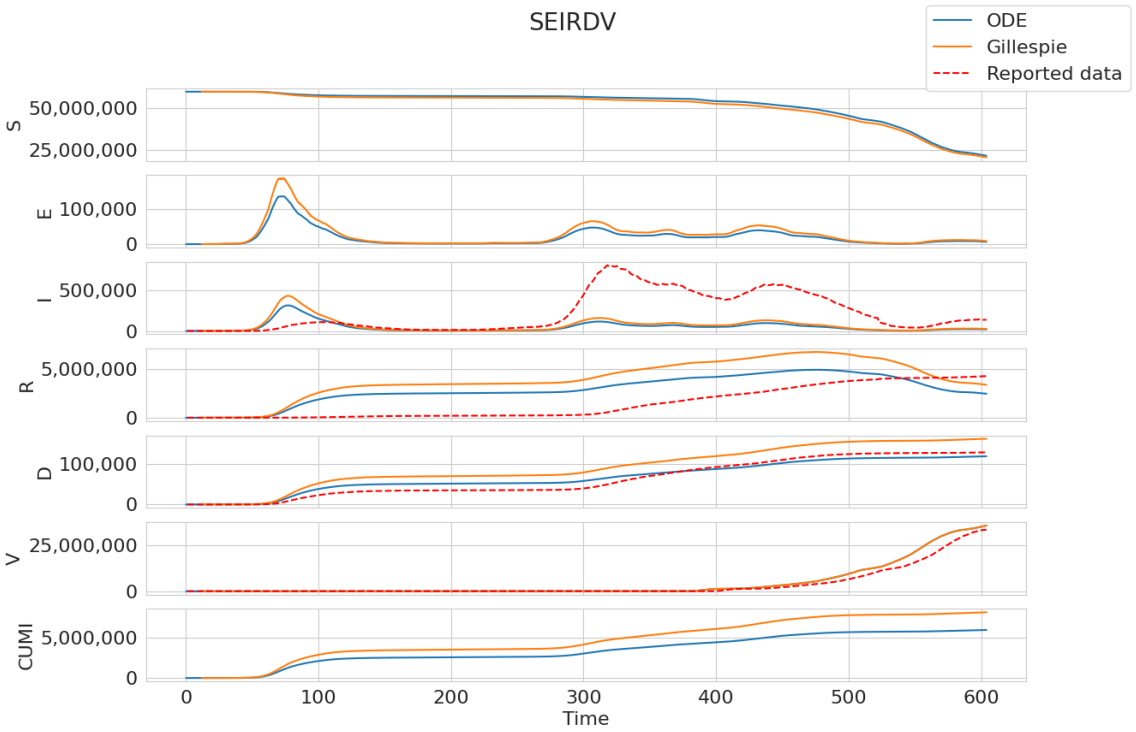


Figure 4.11: SEIRDV simulation results

section 3.2. The model subdivides the population in three classes, namely Young (y), Middle-age (m) and Old (o), that have different related to different transmission and epidemiological parameters. The epidemiological states S, I, R, are all subdivided into the three classes for a total of nine compartments; compartment D is designed only as cumulative compartment to count all the deaths, but it is possible to expand the model to have specific death counts for each class. The simulation spans 604 days and starts with the initial conditions reported in table 4.5. The formula for the force of the infection simply describes the probability of encountering an infected individual from each class. The contact rate names are constructed to represent the class contacts combinations: *contact* + *susceptible class* + *infected class*. Healing and death rate names are also constructed to represent the class combinations: *recovery* + *class*, *death* + *class*.

Extending the SIRDV and SEIRDV models with classes and real data, starting from the incorporation of R_t in the infection force formula, can be achieved by adjusting the transmission rates as described in Section 2.2.3 and inserting them in the configuration file for the model writer program. The BioPEPA formalization of the model can be found in the appendix A. Figure 4.13 shows the schematic representation of the as prepared by the file output `ClassExample.png` of the workbench.

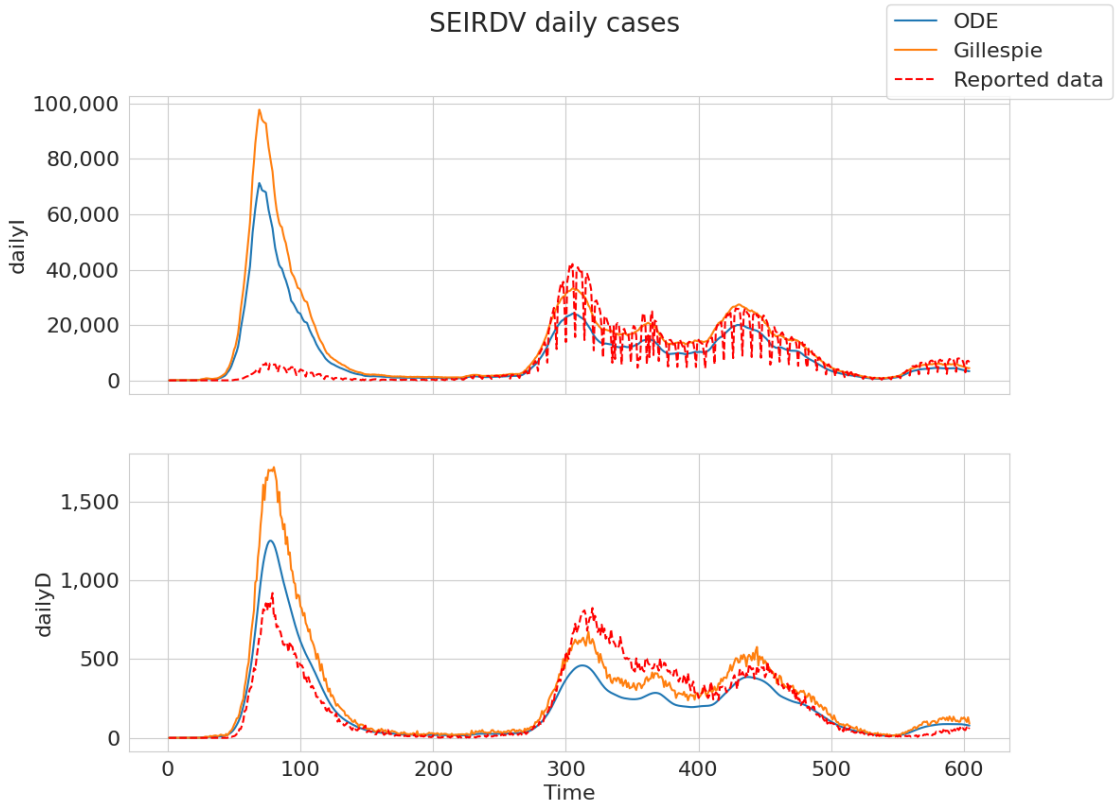


Figure 4.12: SEIRDV simulation inferred daily infections and deaths

Simulation parameters

The parameters are made with mock up data which are derived with a simple educated guess and have no connection with real data of any kind. The rates are created to give the *Young* and *Middleage* classes a higher rate of contacts respect to the *Old* class. The *Young* class population is the less vulnerable to the disease, the *Old* class population is the most vulnerable to the disease, meanwhile the *Middleage* class have a less resilience to the infection but also minor probability of dying. No custom function are used in this model definition.

Simulation results

In figure 4.14 it is possible to see that, as expected from the imposed rate, the three age classes have a different behaviour in each compartments. In particular, the young population (which is double than 'm' and 'o') have the most infected individuals but also the most recovered, meanwhile middle aged individuals have less recovered but also less infected individuals. Lastly the older population have suffered the most causalities as it can be deduced by the higher number of infected individuals than the middle aged class, but an equivalent number of recovered persons.

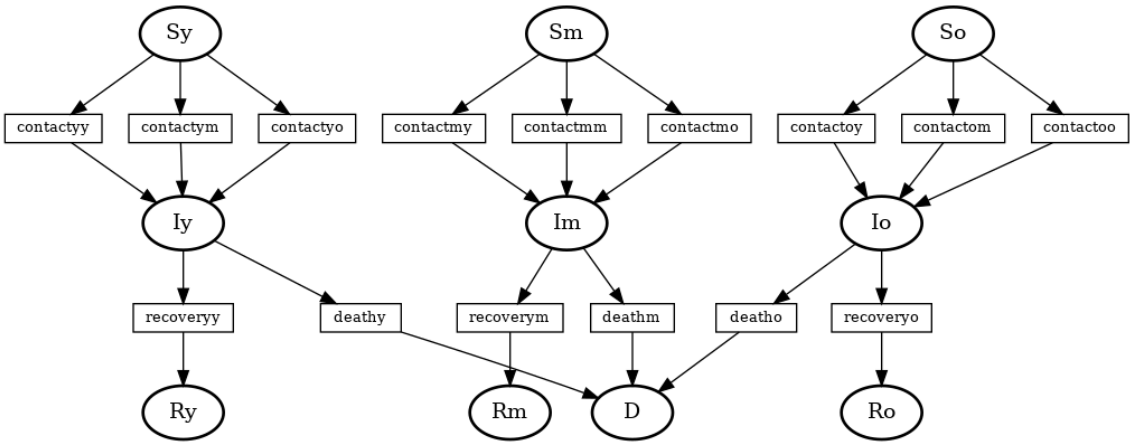


Figure 4.13: Transition scheme of three class BioPEPA model

Parameter	Value
contactyyrate	0.00277
contactymrate	0.02771
contactyorate	0.00277
contactmyrate	0.00249
contactmmrate	0.00388
contactmorate	0.00221
contactoyrate	0.00138
contactomrate	0.02771
contactoorate	0.00055
recoveryyrate	0.005
recoverymrate	0.0035
recoveryorate	0.0025
deathyrate	0.0001
deathmrate	0.00055
deathorate	0.001

Compartment	Initial state
Sy	990
Sm	490
So	500
Iy	10
Im	10
Io	0
Ry	0
Rm	0
Ro	0
D	0

Table 4.5: Parameters values and initial state definition for the SIRD model with the three classes of young (y), middle age (m), and old (o).

4.5 Statistics on the computational times

The computing power and memory requirements for these simulations are within the capabilities of most personal computers. The time taken varies a lot between the ODE and stochastic solvers, but it is acceptable for all the model explored and probably it can be reduced with software upgrades. The elapsed time and the maximum memory occupation are reported in Tables 4.6 and 4.7 for Stochkit (Gillespie solver) and Sundials (ODEs solver), respectively. It must be noted that the possibility of performing the simulations in personal computers might be denied or have unacceptable time performances by some of the tasks discussed in chapter 5, in particular this is highly probable to occur with optimisation algorithms.

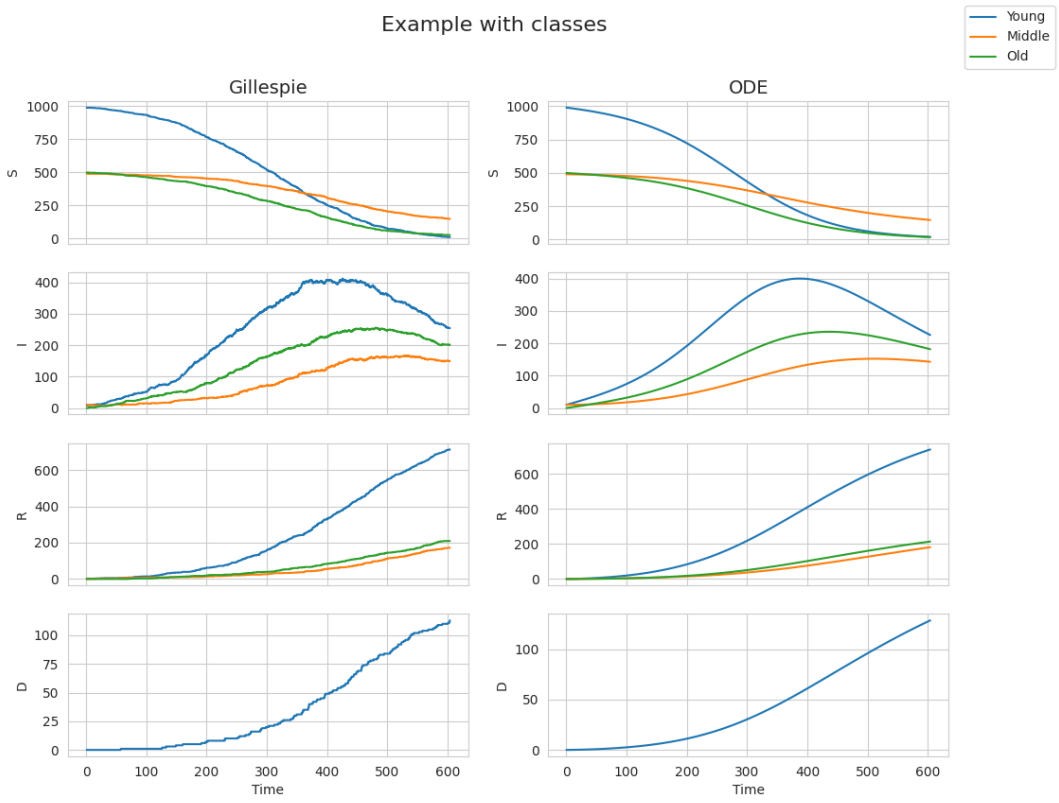


Figure 4.14: Comparison among the simulation results obtained with the Gillespie solver (left) and ODEs solver (right) for the SIRD model with three age classes.

The computer used to run the simulations has the following characteristics: **Intel Core i7-6700 CPU @ 3.40GHz, Cores 4, Threads 8, L1 Data Cache Size 4 x 32 KBytes, L2 Cache Size 4 x 256 KBytes, L3 Cache Size 8192 KBytes, Memory Size 16384 MBytes @ 1066 MHz**.

Simulation	StochKit time taken	StochKit peak memory usage
SIRDV	4 min. 44,58 sec.	12291,8 Mbytes ≈ 75%
SIRDV vimpact 50%	7 min. 4,61 sec.	12290,46 Mbytes ≈ 75%
SIRDV vimpact 66%	4 min. 58,93 sec.	12291,79 Mbytes ≈ 75%
SEIRDV	3 min. 41,66 sec.	6147,59 Mbytes ≈ 37,5%
Class example	0,01 sec.	4,53 Mbytes ≈ 0,028%

Table 4.6: StochKit simulations statistics

Simulation	Sundials time taken	Sundials peak memory usage
SIRDV	0,15 sec.	2,11 Mbytes ≈ 0,01%
SIRDV vimpact 50%	0,13 sec.	2,24 Mbytes ≈ 0,01%
SIRDV vimpact 66%	0,17 sec.	2,16 Mbytes ≈ 0,01%
SEIRDV	0,18 sec.	2,29 Mbytes ≈ 0,01%
Class example	less than 0.01 sec.	2,29 Mbytes ≈ 0,01%

Table 4.7: Sundials simulations statistics

Chapter 5

Conclusions

In this thesis is investigated the use of BioPEPA formal language for the simulation of an epidemic, proposed several extensions to this language which are necessary to consider complex transmission dynamics, and presented the application of different models to the SARS-CoV-2 epidemic in Italy. All simulations results are fairly consistent with reported data without the starting state calibration. Notably, the are not many difference between the accuracy of SIRDV and SEIRDV models, the only important distinction is that the E exposed allows for a buffering which dampens the infection force. This results in a less exponential growth of infection cases, which is more coherent with real life dynamics. Regarding the solvers performances, it is clear that stochastic solver require higher execution times and much more memory capacity than the ODE solver. However the stochastic seems to have a finer grain in miming the epidemic behaviour, which results in a better agreement with recorded data.

The quality of the results for both kinds of simulations is very positive thanks to the inclusion of R_t factor, which confirms that its inclusion in the simulation executions is a good strategy. Without the simulation calibrations it is not possible to appoint a better solver, but it is an interesting possibility to investigate further which simulation technique can perform better with more complex models. The main contribution of this thesis are certainly the program expansion of the Workbench and the creation of the automatic model writer for epidemiological models with many classes. These extensions give the opportunity of comparing stochastic and ODE simulation results of an epidemiological model directly from the same input. Therefore the main objectives of the project can be considered achieved.

Different analysis, elaborations and expansions could not have been performed in this thesis due to lack of time, and are left as possible future research works. These include classes addition in the models to gain insights on subsets of popu-

lation dynamics, which can be easily done taking advantage of the model writer program, considering age, sex, working sector etc. Models should be also expanded with additional states, similarly to [41], for example "hospitalised" or "intensive care" states should be considered in the model to gain more insights for healthcare structure management. Another possible direction of investigation is expanding the workbench together with the custom functions to include the geographical dimension in the models which could be easily done with BioPEPA formalism since it supports the definition of locations. This would allow to model both population classes in conjunction with the geographical behaviour of the epidemic[9]. All these possible developments open up the opportunity of defining an optimization problem, directly in the BioPEPA formalism, which, however, poses many challenges.

The definition of an optimization problem is to find the best solution in the set of all possible solutions and this is object of a vast field of research in computer science and mathematics. In order to introduce optimization directly into BioPEPA it is needed to both expand the formalism of the process algebra and the tool for simulating the models. This would require a considerable amount of work. On the theoretical side it is needed to define and probably proof all the properties to include the formalization of the optimization problem in BioPEPA. Whereas on the practical side it is required to develop the translation of the model into a format describing the mathematical optimization problem and then pass it as input to one of the efficient algorithms that have been developed by the scientific community. In case all of this work is fulfilled, it is important to point out that BioPEPA capabilities would have been greatly expanded but most of these new functionalities would not be available for personal computers as most optimization problems require computational facilities to be resolved in an acceptable amount of time.

Considering the potential of optimization solvers, an interesting application to epidemiological modelling is for example the search for the best vaccination strategy to obtain the minimum number of causalities [10]. Another example could be the study of the most effective restrictions and their combination to minimize the transmission of the disease.

Even if ethical and moral aspects do not strictly belong in the computer science field, it is important to point out that optimized solutions for a problem could pose some serious dilemmas on the ethical side. For example, the vaccination priority could leave behind, for considerable amount of time, some population classes which could introduce some kind of discrimination. Would it be acceptable knowing that it is the best solution that minimize causalities? It is best to leave these considerations to political authorities such that they could find balance between scientific advice for one specific problem with all the other needs and principles of the society.

On a personal note, I desire to point out that the implementation of both tools (Workbench and Eclipse Plugin) for the BioPEPA formalism are quite limited and in need of revision and restructuring in an organic way. Furthermore to the aspects already mentioned in section 3, the BioPEPA Eclipse Plugin and Workbench are missing some important capabilities. For example, in an extension paper of BioPEPA[42], events have been added to the syntax but neither of the two tools are currently supporting events in the simulations.

It is arguable that both tools should be convoluted in a new and comprehensive tool that makes the most of new programming languages and techniques. In particular, since the Eclipse Plugin is not programmable meanwhile the Workbench is not user friendly, the usability would increase greatly if the new tool made available a graphic interface and scripting APIs. The Workbench especially is written in ML, uses libraries written in C, C++, converts the BioPEPA model in other formats, use different graphical tools to plot schematics and line charts (with questionable results) and the execution flow is managed by a terminal script.

Implementing the new tool with OOP patterns in a single library or module, using only one language (like Java or Python) would be the best development option. For example the current logic of the two tools could be incorporated in a single Python module which could take advantage of plenty of libraries for data processing and graphical representation. This would eliminate the need of converting data and metadata in different formats and input files for different tools, reducing the code complexity and taking advantage of higher abstraction when possible.

Appendix A

Appendix

Function added to the Workbench for the StochKit solver

```
1 std::vector<std::vector<double> > datatableValues;
2 std::vector<std::string> datatableHeader;
3
4 int getClassIndex(std::string className)
5 {
6   for (long unsigned int i = 0; i < datatableHeader.size(); ++i)
7   {
8     if (className.compare(datatableHeader.at(i)) == 0)
9       return i;
10  }
11  return 0;
12 }
13
14 double readDatatable(double t, const char *filename, std::string
      className)
15 {
16  /*
17    1 first execution: read the whole file in a vector of Rt values
      . The index of the vector is equal to the daytime of the value
18    2 access vector at index t and get the element
19  */
20  if (datatableValues.empty())
21  {
22    std::string textbuffer;
23    // Read from the text file
24    std::ifstream filestream(filename);
25    std::getline(filestream, textbuffer);  // get header line
26    std::stringstream sstream(textbuffer); // create string stream
      from the string
27    while (sstream.good())
28    {
```

```
29        std::string substr;
30        std::getline(sstream, substr, ','); // get first string
      delimited by comma
31        substr.erase(std::remove(substr.begin(),substr.end(),'\"'),
      substr.end());
32        datatableHeader.push_back(substr);
33      }
34      double readvalue;
35      // Use a while loop together with the getline() function to
      read the file line by line
36      while (std::getline(filestream, textbuffer))
37      {
38        std::vector<std::string> line;
39        std::stringstream sstream(textbuffer); // create string
      stream from the string
40        while (sstream.good())
41        {
42          std::string substr;
43          std::getline(sstream, substr, ','); // get first string
      delimited by comma
44          line.push_back(substr);
45        }
46        std::vector<double> tableRow;
47        for (long unsigned int i = 0; i < line.size(); ++i)
48        {
49          readvalue = atof(line.at(i).c_str());
50          tableRow.push_back(readvalue);
51        }
52        datatableValues.push_back(tableRow);
53        line.clear();
54        tableRow.clear();
55      }
56      // Close the file
57      filestream.close();
58    }
59    double result;
60    int index = getClassIndex(className);
61    if (t <= datatableValues.size())
62      result = datatableValues.at(round(t)).at(index);
63    else
64      result = 0;
65    return result;
66  }
```

Listing A.1: Datatable reading function added to the Workbench

Files to launch the SIRDV model simulations

```
infect = [(I*(S+R+V)*(alpha+gamma1)*placeholder)/(S+I+R+V)];
infectcount = [(I*(S+R+V)*(alpha+gamma1)*placeholder)/(S+I+R+V)];
recover = [gamma1*I];
death = [alpha*I];
vaccinateS = [S*vaccinerate/(S+R)];
vaccinateR = [R*vaccinerate/(S+R)];

S = (infect)<<S + (vaccinateS)<<S;
I = (infect)>>I + (recover)<<I + (death)<<I;
R = (recover)>>R + (vaccinateR)<<R;
D = (death)>>D;
V = (vaccinateS)>>V + (vaccinateR)>>V;

CUMI = (infectcount)>>CUMI;

(S <> CUMI <> I <> R <> D <> V)
```

Listing A.2: SIRDV.biopepa file

```
"placeholder","vaccinerate","gamma1","alpha","S","I","R","D","V","
    CUMI"
1,1,0.196,0.004,59641488,2,0,0,0,2
```

Listing A.3: SIRDV.csv file

```
{
    "biopepa_file_name": "SIRDV",
    "substitutions": [
        {
            "custom_function_c_name": "readRt",
            "custom_function_cpp_name": "readRt",
            "input_parameters": [
                "\"Rt.csv\""
            ],
            "input_time_variable": true,
            "placeholderVariable": "placeholder"
        },
        {
            "custom_function_c_name": "readDatatable",
            "custom_function_cpp_name": "readDatatable",
            "input_parameters": [
                "\"vacciniSIRDV.csv\"",
                "\"n\""
            ],
            "input_time-variable": true,
            "placeholderVariable": "vaccinerate"
        }
```

```
23     ]
24 }
```

Listing A.4: custom_functions_configSIRDV.json file

Files to generate the model for the SIRDV example with three classes

```
1 {
2     "state_number": 4,
3     "states": ["S", "I", "R", "D"],
4     "starting_state": "S",
5     "states_description":
6     {
7         "S": [["contact","I"]],
8         "I": [["recovery","R"],["death","D"]]
9     },
10    "transmission_action": "contact",
11    "transmission_states": ["S", "I"],
12    "internal_classed_states": ["R"],
13    "classless_states":["D"],
14    "model_classes_number": 3,
15    "model_classes": ["y", "m", "o"],
16    "disease_rates_by_class":
17    {
18        "y":{"recovery":0.005,"death":0.0001},
19        "m":{"recovery":0.0035,"death":0.00055},
20        "o":{"recovery":0.0025,"death":0.001}
21    },
22    "total_population": 2000,
23    "initial_state":
24    {
25        "y": [990,10,0,0],
26        "m": [490,10,0,0],
27        "o": [500,0,0,0]
28    }
29 }
```

Listing A.5: JSON configuration file for the model-writer program

```
1 y,m,o
2 0.00005,0.0005,0.00005
3 0.000045,0.00007,0.00004
4 0.000025,0.0005,0.00001
```

Listing A.6: CSV with contact matrix file for the model-writer program

BioPEPA SIRDV model

For the BioPEPA Workbench the SIRDV model described in Section 2 is inserted using the following functional rates:

$$\begin{aligned}
infect &= [(I \times (S+R+V) \times (\alpha+\gamma) \times placeholder)/(S+I+R+V)] \\
infectcount &= [(I \times (S+R+V) \times (\alpha+\gamma) \times placeholder)/(S+I+R+V)] \\
recover &= [\gamma \times I] \\
death &= [\alpha \times I] \\
vaccinateS &= [S \times vaccinerate/(S+R)] \\
vaccinateR &= [R \times vaccinerate/(S+R)]
\end{aligned}$$

Then, the model defines the following compartment definitions:

$$\begin{aligned}
S &= (infect)\downarrow S + (vaccinateS)\downarrow S \\
I &= (infect)\uparrow I + (recover)\downarrow I + (death)\downarrow I \\
R &= (recover)\uparrow R + (vaccinateR)\downarrow R \\
D &= (death)\uparrow D \\
V &= (vaccinateS)\uparrow V + (vaccinateR)\uparrow V \\
CUMI &= (infectcount)\uparrow CUMI
\end{aligned}$$

Where the compartment V counts the number of vaccinated individuals and compartment $CUMI$ counts the new infections (it is useful to show the daily infections).

Finally the system equations are:

$$S \underset{*}{\bowtie} CUMI \underset{*}{\bowtie} I \underset{*}{\bowtie} R \underset{*}{\bowtie} D \underset{*}{\bowtie} V$$

BioPEPA model of SIRDV vaccination impacts simulations

The functional rate definitions are:

$$\begin{aligned}
infect &= [(I \times placeholder)/(S+I+R+V)] \\
infectcount &= [(I \times placeholder)/(S+I+R+V)] \\
recover &= [\gamma \times I] \\
death &= [\alpha \times I] \\
vaccinateS &= [S \times vaccinerate \times percentage/(S+R)] \\
vaccinateR &= [R \times vaccinerate \times percentage/(S+R)]
\end{aligned}$$

SIRDV + CUMI compartments are defined as follows:

$$\begin{aligned} S &= (infect)\downarrow S + (vaccinateS)\downarrow S \\ I &= (infect)\uparrow I + (recover)\downarrow I + (death)\downarrow I \\ R &= (recover)\uparrow R + (vaccinateR)\downarrow R \\ D &= (death)\uparrow D \\ V &= (vaccinateS)\uparrow V + (vaccinateR)\uparrow V \\ CUMI &= (infectcount)\uparrow CUMI \end{aligned}$$

Thus the system equation is:

$$S \underset{*}{\bowtie} CUMI \underset{*}{\bowtie} I \underset{*}{\bowtie} R \underset{*}{\bowtie} D \underset{*}{\bowtie} V$$

BioPEPA SEIRDV model

The functional rate definitions are:

$$\begin{aligned} infect &= [(I \times (S + R + V) \times (\alpha + \gamma) \times placeholder)/(S + E + I + R + V)] \\ infectcount &= [(I \times (S + R + V) \times (\alpha + \gamma) \times placeholder)/(S + E + I + R + V)] \\ incubate &= [\delta \times E] \\ recover &= [\gamma \times I] \\ death &= [\alpha \times I] \\ vaccinateS &= [S \times vaccinerate/(S + R)] \\ vaccinateR &= [R \times vaccinerate/(S + R)] \end{aligned}$$

SEIRDV + CUMI compartments are defined as follows:

$$\begin{aligned} S &= (infect)\downarrow S + (vaccinateS)\downarrow S \\ E &= (infect)\uparrow E + (incubate)\downarrow E \\ I &= (infect) \oplus +(incubate)\uparrow I + (recover)\downarrow I + (death)\downarrow I \\ R &= (recover)\uparrow R + (vaccinateR)\downarrow R \\ D &= (death)\uparrow D \\ V &= (vaccinateS)\uparrow V + (vaccinateR)\uparrow V \\ CUMI &= (infectcount)\uparrow CUMI \end{aligned}$$

Lastly the system equation is:

$$S \underset{*}{\bowtie} E \underset{*}{\bowtie} CUMI \underset{*}{\bowtie} I \underset{*}{\bowtie} R \underset{*}{\bowtie} D \underset{*}{\bowtie} V$$

BioPEPA model example with 3 classes

The functional rate definitions are:

$$\begin{aligned}
contactyy &= [(contactyyrate/(Ry + Iy + Sy)) \times Sy \times Iy] \\
contactym &= [(contactymrate/(Rm + Im + Sm)) \times Sy \times Im] \\
contactyo &= [(contactyorate/(Ro + Io + So)) \times Sy \times Io] \\
contactmy &= [(contactmyrate/(Ry + Iy + Sy)) \times Sm \times Iy] \\
contactmm &= [(contactmmrate/(Rm + Im + Sm)) \times Sm \times Im] \\
contactmo &= [(contactmorate/(Ro + Io + So)) \times Sm \times Io] \\
contactoy &= [(contactoyrate/(Ry + Iy + Sy)) \times So \times Iy] \\
contactom &= [(contactomrate/(Rm + Im + Sm)) \times So \times Im] \\
contactoo &= [(contactoorate/(Ro + Io + So)) \times So \times Io] \\
recoveryy &= [recoveryyrate \times Iy] \\
recoverym &= [recoverymrate \times Im] \\
recoveryo &= [recoveryorate \times Io] \\
deathy &= [deathyrate \times Iy] \\
deathm &= [deathmrate \times Im] \\
deatho &= [deathorate \times Io]
\end{aligned}$$

Then the compartment are defined as:

$$\begin{aligned}
Sy &= (contactyy, 1)\downarrow Sy + (contactym, 1)\downarrow Sy + (contactyo, 1)\downarrow Sy \\
Sm &= (contactmy, 1)\downarrow Sm + (contactmm, 1)\downarrow Sm + (contactmo, 1)\downarrow Sm \\
So &= (contactoy, 1)\downarrow So + (contactom, 1)\downarrow So + (contactoo, 1)\downarrow So \\
Iy &= (contactyy, 1)\uparrow Iy + (contactym, 1)\uparrow Iy + (contactyo, 1)\uparrow Iy + (contactmy, 1)\oplus \\
&\quad +(contactoy, 1) \oplus +(recoveryy, 1)\downarrow Iy + (deathy, 1)\downarrow Iy \\
Im &= (contactym, 1) \oplus +(contactmy, 1)\uparrow Im + (contactmm, 1)\uparrow Im + (contactmo, 1)\uparrow Im \\
&\quad +(contactom, 1) \oplus +(recoverym, 1)\downarrow Im + (deathm, 1)\downarrow Im \\
Io &= (contactyo, 1) \oplus +(contactmo, 1) \oplus +(contactoy, 1)\uparrow Io + (contactom, 1)\uparrow Io \\
&\quad +(contactoo, 1)\uparrow Io + (recoveryo, 1)\downarrow Io + (deatho, 1)\downarrow Io \\
Ry &= (recoveryy, 1)\uparrow Ry \\
Rm &= (recoverym, 1)\uparrow Rm \\
Ro &= (recoveryo, 1)\uparrow Ro \\
D &= (deathy, 1)\uparrow D + (deathm, 1)\uparrow D + (deatho, 1)\uparrow D
\end{aligned}$$

Leading to the complete system definition:

$$Sy \underset{*}{\bowtie} Sm \underset{*}{\bowtie} So \underset{*}{\bowtie} Iy \underset{*}{\bowtie} Im \underset{*}{\bowtie} Io \underset{*}{\bowtie} Ry \underset{*}{\bowtie} Rm \underset{*}{\bowtie} Ro \underset{*}{\bowtie} D$$

Bibliography

[1] *Coronavirus, Climate Change, and the Environment A Conversation on COVID-19 with Dr. Aaron Bernstein, Director of Harvard Chan C-CHANGE*. [Online; accessed 26-September-2022]. URL: https://www.hsph.harvard.edu/c-change/subtopics/coronavirus-and-climate-change/.

[2] *Climate change is making hundreds of diseases much worse*. [Online; accessed 26-September-2022]. URL: https://www.nature.com/articles/d41586-022-02167-z#.

[3] Camilo Mora et al. "Over half of known human pathogenic diseases can be aggravated by climate change". In: *Nature Climate Change* 12 (Aug. 2022). DOI: 10.1038/s41558-022-01426-1.

[4] Colin Carlson et al. "Climate change increases cross-species viral transmission risk". In: *Nature* 607 (July 2022), pp. 1–1. DOI: 10.1038/s41586-022-04788-w.

[5] C. Drew Harvell et al. "Climate Warming and Disease Risks for Terrestrial and Marine Biota". In: *Science* 296.5576 (2002), pp. 2158–2162. DOI: 10.1126/science.1063699. eprint: https://www.science.org/doi/pdf/10.1126/science.1063699. URL: https://www.science.org/doi/abs/10.1126/science.1063699.

[6] *The mystery viruses far worse than flu*. [Online; accessed 26-September-2022]. URL: https://www.bbc.com/future/article/20181101-the-mystery-viruses-far-worse-than-flu.

[7] *WHO Covid-19 media speech from pandemic status declaration*. [Online; accessed 16-September-2022]. URL: https://www.who.int/director-general/speeches/detail/who-director-general-s-opening-remarks-at-the-media-briefing-on-covid-19---11-march-2020.

[8] European Centre for Disease Prevention and Control. *Overview of the implementation of COVID-19 vaccination strategies and vaccine deployment plans in the EU/EEA*. Feb. 2021.

[9] Joseph Chadi Lemaitre et al. "Optimal control of the spatial allocation of COVID-19 vaccines: Italy as a case study". In: *PLOS Computational Biology* 18 (July 2022), pp. 1–20. DOI: 10.1371/journal.pcbi.1010237. URL: https://doi.org/10.1371/journal.pcbi.1010237.

[10] Laura Matrajt et al. "Vaccine optimization for COVID-19: Who to vaccinate first?" In: *Science Advances* 7.6 (2021), eabf1374. DOI: 10.1126/sciadv.abf1374. URL: https://www.science.org/doi/abs/10.1126/sciadv.abf1374.

[11] Theodore Tulchinsky. "John Snow, Cholera, the Broad Street Pump; Waterborne Diseases Then and Now". In: Jan. 2018, pp. 77–99. ISBN: 9780128045718. DOI: 10.1016/B978-0-12-804571-8.00017-2.

[12] Jin Wang and Chairat Modnak. "Modeling cholera dynamics with controls". In: *Canadian applied mathematics quarterly* 19.3 (2011), pp. 255–273.

[13] Danilo Cereda et al. "The early phase of the COVID-19 epidemic in Lombardy, Italy". In: *Epidemics* 37 (2021), p. 100528. ISSN: 1755-4365. DOI: https://doi.org/10.1016/j.epidem.2021.100528. URL: https://www.sciencedirect.com/science/article/pii/S1755436521000724.

[14] Marino Gatto et al. "Spread and dynamics of the COVID-19 epidemic in Italy: Effects of emergency containment measures". In: *Proceedings of the National Academy of Sciences* 117.19 (2020), pp. 10484–10491. DOI: 10.1073/pnas.2004978117. eprint: https://www.pnas.org/doi/pdf/10.1073/pnas.2004978117. URL: https://www.pnas.org/doi/abs/10.1073/pnas.2004978117.

[15] Enrico Bertuzzo et al. "The geography of COVID-19 spread in Italy and implications for the relaxation of confinement measures". In: *Nature Communications* 11 (Aug. 2020). DOI: 10.1038/s41467-020-18050-2.

[16] Valentina Marziano et al. "Retrospective analysis of the Italian exit strategy from COVID-19 lockdown". In: *Proceedings of the National Academy of Sciences* 118.4 (2021), e2019617118. DOI: 10.1073/pnas.2019617118. eprint: https://www.pnas.org/doi/pdf/10.1073/pnas.2019617118. URL: https://www.pnas.org/doi/abs/10.1073/pnas.2019617118.

[17] Nicola Parolini et al. "Modelling the COVID-19 epidemic and the vaccination campaign in Italy by the SUIHTER model". In: *Infectious Disease Modelling* 7.2 (2022), pp. 45–63. ISSN: 2468-0427. DOI: https://doi.org/10.1016/j.idm.2022.03.002. URL: https://www.sciencedirect.com/science/article/pii/S2468042722000100.

[18] Federica Ciocchetta and Jane Hillston. "Bio-PEPA: A framework for the modelling and analysis of biological systems". In: *Theoretical Computer Science* 410.33 (2009). Concurrent Systems Biology: To Nadia Busi (1968–2007), pp. 3065–3084. ISSN: 0304-3975. DOI: https://doi.org/10.1016/j.tcs.2009.02.037. URL: https://www.sciencedirect.com/science/article/pii/S0304397509001662.

[19] *BioPEPA Workbench official site*. [Online; accessed 01-September-2022]. URL: https://homepages.inf.ed.ac.uk/stg/software/biopepa/.

[20] *BioPEPA Eclipse Plugin official site*. [Online; accessed 01-September-2022]. URL: https://homepages.inf.ed.ac.uk/stg/research/biopepa/eclipse/manual/manual_new/index.html.

[21] *StochKit official site*. [Online; accessed 26-September-2022]. URL: https://cse.cs.ucsb.edu/.

[22] *Sundials official site*. [Online; accessed 26-September-2022]. URL: https://computing.llnl.gov/projects/sundials.

[23] Ross Beckley Cametria Weatherspoon Michael Alexander Marissa Chandler Anthony Johnson and Ghan S Bhatt. "Modeling epidemics with differential equations". In: (2013). [Online; accessed 26-September-2022]. URL: https://www.tnstate.edu/mathematics/mathreu/filesreu/GroupProjectSIR.pdf.

[24] Linda J.S. Allen. "A primer on stochastic epidemic models: Formulation, numerical simulation, and analysis". In: *Infectious Disease Modelling* 2.2 (2017), pp. 128–142. ISSN: 2468-0427. DOI: https://doi.org/10.1016/j.idm.2017.03.001. URL: https://www.sciencedirect.com/science/article/pii/S2468042716300495.

[25] Nicolas Privault. "Understanding markov chains". In: *Examples and Applications, Publisher Springer-Verlag Singapore* 357 (2013), p. 358.

[26] Daniel T. Gillespie. "Exact stochastic simulation of coupled chemical reactions". In: *The Journal of Physical Chemistry* 81.25 (1977), pp. 2340–2361. DOI: 10.1021/j100540a008. URL: https://doi.org/10.1021/j100540a008.

[27] A. Rathinasamy, M. Chinnadurai, and S. Athithan. "Analysis of exact solution of stochastic sex-structured HIV/AIDS epidemic model with effect of screening of infectives". In: *Mathematics and Computers in Simulation* 179 (2021), pp. 213–237. ISSN: 0378-4754. DOI: https://doi.org/10.1016/j.matcom.2020.08.017. URL: https://www.sciencedirect.com/science/article/pii/S0378475420302846.

[28] Warren Hall. “Epidemiological models with age structure, proportionate mixing, and cross-immunity”. In: (1989).

[29] J. L. Manthey, A. Y. Aidoo, and K. Y. Ward. “Campus drinking: an epidemiological model”. In: *Journal of Biological Dynamics* 2.3 (2008). PMID: 22876873, pp. 346–356. DOI: 10.1080/17513750801911169. URL: https://doi.org/10.1080/17513750801911169.

[30] Md. Arif Billah, Md. Mamun Miah, and Md. Nuruzzaman Khan. “Reproductive number of coronavirus: A systematic review and meta-analysis based on global level evidence”. In: *PLOS ONE* 15.11 (Nov. 2020), pp. 1–17. DOI: 10.1371/journal.pone.0242128. URL: https://doi.org/10.1371/journal.pone.0242128.

[31] Odo Diekmann, J.A.P. Heesterbeek, and M.G. Roberts. “The construction of next-generation matrices for compartmental epidemic models”. In: *Journal of the Royal Society, Interface / the Royal Society* 7 (Nov. 2009), pp. 873–85. DOI: 10.1098/rsif.2009.0386.

[32] *PEPA official site*. [Online; accessed 01-September-2022]. URL: https://www.dcs.ed.ac.uk/pepa/.

[33] *PRISM official site*. [Online; accessed 01-September-2022]. URL: https://www.prismmodelchecker.org/.

[34] Federica Ciocchetta and Jane Hillston. “Bio-PEPA for Epidemiological Models”. In: *Electronic Notes in Theoretical Computer Science* 261 (2010). Proceedings of the Fourth International Workshop on the Practical Application of Stochastic Modelling (PASM 2009), pp. 43–69. ISSN: 1571-0661. DOI: https://doi.org/10.1016/j.entcs.2010.01.005. URL: https://www.sciencedirect.com/science/article/pii/S157106611000006X.

[35] *Eclipse IDE official site*. [Online; accessed 01-September-2022]. URL: https://www.eclipse.org/ide/.

[36] *BioPEPA official site*. [Online; accessed 01-September-2022]. URL: https://homepages.inf.ed.ac.uk/jeh/Bio-PEPA/biopepa.html.

[37] *Epidemiological code repository*. [Online; accessed 30-September-2022]. URL: https://github.com/RPComputer/Epidemiological-BioPEPA-Model.

[38] *Dizzy simulator official site*. [Online; accessed 01-September-2022]. URL: https://homepages.inf.ed.ac.uk/stg/software/Dizzy/.

[39] *Istituto Superiore di Sanità*. [Online; accessed 01-September-2022]. URL: https://www.epicentro.iss.it/coronavirus/sars-cov-2-sorveglianza-dati.

[40] *Istituto Superiore di Sanità calcolo Rt*. [Online; accessed 01-September-2022]. URL: https://www.epicentro.iss.it/coronavirus/open-data/rt.pdf.

[41] Marco Gribaudo, Mauro Iacono, and Daniele Manini. "COVID-19 Spatial Diffusion: A Markovian Agent-Based Model". In: *Mathematics* 9.5 (2021). ISSN: 2227-7390. URL: https://www.mdpi.com/2227-7390/9/5/485.

[42] Federica Ciocchetta. "Bio-PEPA with Events". In: *Transactions on Computational Systems Biology XI*. Ed. by Corrado Priami, Ralph-Johan Back, and Ion Petre. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 45–68. DOI: 10.1007/978-3-642-04186-0_3. URL: https://doi.org/10.1007/978-3-642-04186-0_3.

List of Figures

List of Tables